91602
new 82-10

LEGAL NOTICE

This report was prepared by Ontario Hydro and, while effort has been made to assure the accuracy and applicability of the material contained herein, neither Ontario Hydro nor any person acting on its behalf:

(a) makes any warranty, express or implied with respect to the use of any information, apparatus, method or process disclosed in this report or that such use may not infringe privately owned rights; or

(b) assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method or process disclosed in this report.

ontario hydro
research division

COMPLIMENTARY COPY
Please Credit Ontario Hydro Research if Reproduced in Whole or in Part

EXPECT - EXPERT SYSTEM FOR EDDY CURRENT TESTING

Research Report No 87-173-K

PART I

S.B. Nickerson
Theoretical Physicist
Science Section
Electrical Research Department

ABSTRACT

This report documents an investigation into the feasibility of developing a software aid to assist in analyzing the signals from eddy current probes. These probes are used for the detection of flaws in heat exchanger tubes. The major advantage to using eddy current probes is its speed and the major disadvantage is the occasional difficulty in interpretation of the signals. The philosophy behind the approach taken in this investigation (a rule based expert system) is to duplicate in software as much as possible the expertise that has developed over the years in interpretation of the signals. A prototype expert system (EXPECT - Expert System for Eddy Current Testing) has been developed with a rule base restricted to a subset of the total domain. It has been tested with data from twenty heat exchanger tubes from Pickering Nuclear Generating Station Units 5 and 6. There has been considerable success in the interpretation of the simpler signals by EXPECT and some mixed success and good prospects for the more complicated signals. Recommendations are given for the work needed to complete the project.

job	file	date	report no
16-203	819.322	August 12, 1988	87-173-K

EXPECT - EXPERT SYSTEM FOR EDDY CURRENT TESTING

PART I

TABLE OF CONTENTS

	Page
TABLE OF CONTENTS	i,ii,iii
1. INTRODUCTION	1
2. DESCRIPTION OF EXPECT AND GUIDE TO THE PROGRAM LISTINGS	3
2.1 Introduction	3
2.2 Description of the EXPECT Program	3
2.2.1 Event Extraction from the X - Y Data	3
2.2.2 The Data Structure	4
2.2.3 The Control Structure	4
2.3 The Addition of New Rules to EXPECT	6
2.4 How to Run EXPECT	7
2.5 Guide to Appendixes E - I	7
2.5.1 Guide to Appendix E	8
2.5.2 Guide to Appendix F	8
2.5.3 Guide to Appendix G	9
2.5.4 Guide to Appendix H	10
2.5.5 Guide to Appendix I	10
2.6 Hardware and Software	11
3. A CATALOGUE OF EDDY CURRENT SIGNAL INTERPRETATION EXPERTISE	12
3.1 Overview	12
3.2 Interviews with Eddy Current Experts	15
3.3 Event Analysis	17
3.3.1 Events Due to Single Causes	17
3.3.2 Composite Signals	18
3.3.2.1 Recognize-and-analyze Approach to Decomposing the Composite Signals	21
3.3.2.2 Analyst with Adder Approach to Decomposing the Composite Signals	22
3.3.2.3 Least Squares Fit Approach to Decomposing the Composite Signals	23
3.3.2.4 Analyst Plus Least Squares Fit Approach to Decomposing the Composite Signals	23
3.3.2.5 Recognize-and-analyze Plus Least Squares Fit Approach to Decomposing the Composite Signals	23
3.3.2.6 The Problem with Non-uniqueness in the Decomposition	23

89-02644/

TABLE OF CONTENTS (Cont'd)

	Page
3.4 Types of Eddy Current Signals and Rules of Analysis	24
3.4.1 Support Plate	25
3.4.2 Tube Sheet	35
3.4.3 U-bend Drift	35
3.4.4 Wall Loss	36
3.4.5 Whalies	38
3.4.6 Mechanical Deformation	39
3.4.7 Deposit	41
3.4.8 Resistivity Variations	41
3.4.9 Magnetic Permeability Changes	41
3.4.10 Intergranular Attack	41
3.4.11 Support Plate Plus Wall Loss	41
3.4.12 Fretting at the Support Plate	42
3.4.13 Tube Sheet Plus Wall Loss	45
3.4.14 Support Plate Plus Mechanical Deformation	45
3.4.15 Pit Within a Pit	45
3.4.16 Dents	45
3.4.17 Other Combinations	47
3.4.18 Change in Balance Point	47
3.4.19 Probe Wobble	47
3.4.20 Characteristics of ANDES	47
3.4.21 Saturated Signal	48
3.4.22 Die Chatter	49
3.5 EXPECT Variables	50
4. RESULTS OF USING EXPECT WITH TUBE DATA	56
5. CONCLUSIONS AND RECOMMENDATIONS	67
APPENDIX A EXPECT VARIABLES FOR TUBE T-1-11	
APPENDIX B DATA STRUCTURE FOR TUBE T-1-11	
APPENDIX C USING EXPECT WITH TUBES OTHER THAN TUBE T-1-11	
APPENDIX D CNS ANALYSIS OF THE TUBES T-... OF TABLE 4.1	

EXPECT - EXPERT SYSTEM FOR EDDY CURRENT TESTING

PART II

This consists of Program Listings and is available upon request.

TABLE OF CONTENTS

APPENDIX E UTILITY FILES

APPENDIX F EXPECT FILES

APPENDIX G RULE FILES

APPENDIX H ENGLISH TRANSLATION OF THE RULES

APPENDIX I INFERENCE ENGINE FILES

APPENDIX J EXPECT VARIABLES FOR TUBES T-CAL-0, T-1-22R, AND T-2-6

ontario hydro
research division

To: Mr. J. Graham
Manager, Inspection and Maintenance
Technical and Training Services A7 C6

EXPECT - EXPERT SYSTEM FOR EDDY CURRENT TESTING

1. INTRODUCTION

Ontario Hydro does enough in-service tube inspections to keep fifteen licensed analysts busy all year. The biggest problem is with corrosion. There is high incentive for this activity since a heat exchanger tube failure in an operating nuclear plant can cost two weeks down time - at about $250 000 per day.

The objective of an inspection is to identify the depth of any defects in a tube. The depth is an indication of how close the tube is to failure. If the tube defect is too deep then the tube is plugged and not used again.

Tubes are inspected by driving an eddy current probe through the tube. The probe (which is basically a magnetic field coil) when excited by an AC current, induces eddy currents in the surrounding tube which in turn produce their own magnetic field. This induced field changes the impedance of the probe and results the signals such as those marked X and Y (strip chart signals) at the bottom of Figure 3.8. If the tube were completely featureless then both of the signals marked X and Y would be straight lines. However, in a real tube there are features such as support plates, cracks, pits etc. which cause changes in the impedance of the probe and produce the peaks shown in the strip chart. The X and Y values are plotted against each other with each peak plotted separately as shown at the top of Figure 3.8. These characteristic figures are visually inspected by analysts and a diagnosis of the tube is made.

A typical inspection time for a 55 foot heat exchanger tube is 1. to 1.5 minutes. Such a tube may give about 40 to 50 indications (called events - the figures at the top of Figure 3.8), the vast majority of which will be due to normal tube support signals and not be the indication of any problems. Since a shift will typically inspect about 500 tubes, this results in 20 000 to 25 000 events to examine per shift.

The software system presently being used is called ANDES. It takes the signals from the probes, stores then on an optical disk and produces an output, such as is shown in Figure 3.8, on a fast thermal printer. This output is then examined by the analyst.

The interpretation is a highly skilled activity because signals that mean the same thing can vary considerably and signals that look almost the same (to the untrained eye) can mean entirely different things. Signals can be generated by support plates (several different kinds), by deposits of magnetite or copper and by a range of defects that include pits, cracks, erosion, wastage corrosion, fretting, dents and intergranular attack (corrosion along grain boundaries).

Signals are also generated by geometry variations in the tube, the ends of the tube and some slight imperfections due to the manufacturing process. As well defects can occur in combination with each other or with a normal signal (composite signals). All of the above will differ according to the type of probe (two common types) and the type of material (six common types).

In order to focus this study it was restricted to the absolute probe and one type of tube - the Incoloy 800 moderator heat exchanger - 1/2" tubing.

The motivation for the development of EXPECT (Expert System for Eddy Current Testing) is to improve consistency in the interpretation of the signals. Therefore, in practice EXPECT will function as an analyst's helper and not supplant the analyst's role. Much of the skill involved takes years to accumulate and it is difficult for the analysts to share their experiences with complicated cases. It is anticipated that EXPECT will provide a mechanism for doing this.

When a signal is analyzed by the human analyst the interpretation is not based on the hundreds of separate data points, but rather it is based on a relatively small number of features of the signal's shape and size. For example, in Figure 3.1 the events marked "sp" are normal support plate signals. The features looked for are: the angle of a line from balance point (marked by a cross) to the tip (marked by a broken X) is within 20° of the - X axis; the initial angle at the balance point is in the 1st or 2nd quadrant, the signal going out to the tip from the balance point is the same as the signal returning, the signal has an "S" shape, and the length and angle of the figure is close to the average for normal support plate signals in the same tube.

EXPECT performs an analysis in the same way as the human analyst. Each event is analyzed for a predetermined set of features such as those mentioned above. The set of features is then passed through a rule base which results in an analysis. The rules in the rule base are written in terms of the same features that the analysts use, thus taking advantage of the years of accumulated knowledge in interpretation. There are however some problems with this approach as discussed in Chapter 3.

The advantage of processing the signals this way is that the rules are written in a form that is easy to debug, easy for the analyst to read (i.e. it is not buried in code) and makes it easy to add new rules. This last item is particularly important because the rule base is continually being increased (at a rate of about four new interpretations per year) and is discussed in more detail in Chapter 2.

Chapter 2 is a description of the EXPECT program itself, how it works, the data and control structures, how to add new rules to EXPECT (independent of EXPECT's author), how to run the program, guides to the program listings in the appendixes and a discussion of the hardware and software used for EXPECT. Because the program listings are so long, they are part of a separate document.

Chapter 3 is the longest chapter and documents the knowledge base of interpretation expertise and the rules that were developed. There is also a discussion of those aspects (how to analyze composite signals and the problem with non-uniqueness in decomposition of the signals) for which more research is required.

Analysis and timing results for EXPECT with ANDES data from several tubes are given in Chapter 4.

Chapter 5 summarizes the conclusions and outlines the steps needed to develop a complete working system.

2. DESCRIPTION OF EXPECT AND GUIDE TO THE PROGRAM LISTINGS

2.1 Introduction

This chapter is a description of the computer program and a few notes on its evolution. There are also sections on how to add new rules to EXPECT without program surgery, how to run EXPECT, a guide to the appendixes containing the program listings (E to I) and some notes on the software and hardware used. The guides to the appendixes do not have much detail since the programs are well documented with comment statements. In addition to the EXPECT program itself there are a few other procedures that were developed during the course of this project. For the sake of completeness these are also listed in the appendixes and mentioned in the guide.

Although the rule system and control structure are discussed, there is no discussion of the details of the individual rules themselves in this chapter. This is done in Chapter 3, amidst a presentation of the signal interpretation knowledge base.

Warning: some of the material in this chapter requires knowledge of LISP.

2.2 Description of the EXPECT Program

This section has three subsections: a discussion of the event extraction procedures (the part of EXPECT that duplicates ANDES), a description of how the data for each tube is stored, and a description of the flow of control in the program.

2.2.1 Event Extraction from the X - Y Data

Raw data from the ANDES program comes with a 184 byte header followed by X - Y pairs of data (one value per byte, stored in two's complement format). The header contains information about the tube, when the data was taken and the size of the file. The raw ANDES data is stored on a disk under a name based on the ANDES tube_number in the header (bytes 2 - 13), with no extension. Table 4.1 lists the file names of the tubes that were provided for this study. The T- prefix was added to the name to indicate that the tube was in the second group.

The files containing the raw data are read by the procedure EVENT-FILL (page F-21), and the peaks in the data were separated from the inter-peak background. This is documented by the comment statements for the procedures in file EVENT.LSP (starting at page F-21). The X - Y pairs and the information extracted from them are stored in a LISP DEFSTRUCT structure - see the following sections.

Any working system will be integrated with ANDES and use its event extraction software. Since this is an exploratory phase it was faster to write a (non-optimized) event extractor than to learn to use ANDES.

Refer to Chapter 4 for the performance of EXPECT in the extraction of peaks from the raw ANDES data.

2.2.2 The Data Structure

All of the data for each tube, both the raw data (the X - Y pairs representing the peaks), the calculated data and the analysis are contained in a LISP DEFSTRUCT data structure. An example for the tube T-1-11 (see Table 4.1) is given in Appendix B.

LISP has fast primitives for accessing the data stored in a DEFSTRUCT structure and allows the definition of procedures (GET-VALUE and PUT-VALUE, page F-3) for accessing the value of a variable without having to know where the value is located (for example in FORTRAN a location might be the fifth position in common statement xyz). All that is needed is the variable's name and the value can be obtained by a procedure call from anywhere in the program. This has the same effect as declaring the variable to be global but with an important distinction (other than the danger of name clashes): a global variable would be available anywhere within the LISP environment but not outside it. During the next phase it is planned to link EXPECT to ANDES, which is written in C, and it is anticipated that, at some point, parts of EXPECT will be written in C (for efficiency). The particular LISP being used (Gold Hill's GCLISP) has a direct interface to C (Lattice C or Microsoft C) so that LISP can directly call C procedures and vice versa. This means that ANDES and any other C program can directly access the values stored in the DEFSTRUCT structures.

This ability to directly access the program variables also means that it is possible to add new rules without doing surgery to the program - see Section 2.3.

The definition of the DEFSTRUCT structures and the procedures for accessing the stored values are in file STRUCT.LSP (page F-3).

Once the peaks have been separated and the data has been analyzed, it is stored in a file with the same name as the raw data file but with an S extension. A printout of the file T-1-11.S (with linefeeds added) is shown in Appendix B.

2.2.3 The Control Structure

To initiate EXPECT use the command LD to load the the file LOADER.LSP (page E-2) for interpreted programs or LOAFER.LSP (page E-3) for compiled programs. To run EXPECT use use the procedure AT (page F-2). AT calls ANALYZE-TUBE (page F-2) which calls FRONT-AND-CENTER (page F-15). FRONT-AND-CENTER will check to see if a data file with an S extension exists for the tube specified and if so it reads in the file and displays the data. If no such file exists or the data in the DEFSTRUCT structure is incomplete, FRONT-AND-CENTER calculates the missing data and displays the results.

The principal DEFSTRUCT structure has the name TUBE (see the 2nd line of Appendix B and page F-13). Each slot in the TUBE structure (except the first which is a list of the rest of the slots) contains another DEFSTRUCT structure. The control structure of EXPECT is quite simple: it fills the DEFSTRUCT structure in each slot of the TUBE structure in order, from the second to the last. Each slot uses information from the previous slot and the last contains the analysis of the tube. When all the information needed is calculated a copy of the structure is stored. This sequential filling of the slots of TUBE takes place from FRONT-AND-CENTER.

After control is passed back to ANALYZE-TUBE another procedure is called, DISPLAY-RESULTS, which presents displays such as shown in Figure 3.1 and is discussed further in Section 2.4.

The procedures to fill the slots in the TUBE structure, and their file and Appendix locations are shown in Table 2.1. The indicated file contains all the subordinate procedures to the appropriate ...-FILL procedure (see Table).

Table 2.1 Information about the slots of the TUBE DEFSTRUCT structure. The procedure referred to is the name of the procedure that fills the indicated slot. The page number is the page number in the Appendix where the procedure and its supporting procedures are listed. File name is the name of the file where the procedures are located (see the global variable FILE-LIST page E-1). Slot contents is a general description of the information contained in the indicated slot.

Slot Name	Procedure Name	Page Number of Procedure	File Name of Procedure	Slot Contents
SLOT-LIST	none	none	none	A list of the rest of the slots
PARAMETERS	PARAMETERS-FILL	F-18	PARAMTRS.LSP	Various adjustable parameters used throughout EXPECT.
INFO	INFO-FILL	F-19	INFO.LSP	ANDES header information
EVENT	EVENT-FILL	F-21	EVENT.LSP	The peaks in the X - Y data separated from the background and the value of the local base
LIMITS	LIMITS-FILL	F-27	LIMITS.LSP	Maximum and minimum X and Y values - used for display
CALCULATED-PROPERTIES	CALCULATED-PROPERTIES-FILL	F-28	CALCPROP.LSP	All the basic calculated properties such as TIP-INDEX (see page for F-8 for a complete list).
NOSP	NOSP-FILL	F-33	NOSP.LSP	The properties needed to identify an event as a normal support plate
NON-NOSP	NON-NOSP-FILL	F-38	NON-NOSP.LSP	The properties needed to identify any type of signal other than a normal support plate
ANALYSIS	ANALYSIS-FILL	F-42	ANALYSIS.LSP	Contains the analysis of the tube

2.3 The Addition of New Rules to EXPECT

It is important that the user be able to add new rules to EXPECT, independent of EXPECT's author, as new interpretations become available. EXPECT would rapidly become useless if it could not be kept up to date.

The most desirable goal would be for a someone unsophisticated in computing to be able to add new rules at will. A less desirable goal (and the one achieved in the present version of EXPECT) is for a person, who is sophisticated in computing, to be able to add new rules without having to perform surgery on EXPECT. To be able to do this involves access to the variables stored in the TUBE DEFSTRUCT structure (the programmer has to know what is available) and the ability to add a new rule. EXPECT allows both of these to occur; the former is discussed in Section 2.2.2 and the latter is explained below.

To add a new rule to EXPECT involves two steps: add the name of the file containing the rule to the variable RULE-LIST (see page E-1) and place the new rule in the file in the format shown in Appendix G.

The format of the rule is basically a list of predicates that test properties of the signal. If all the predicates are true then the signal has passed the rule, the analysis procedure is called and the analysis is written to the ANALYSIS slot of TUBE. The analysis procedure may just return a string (such as "sp" to indicate a normal support plate) or it may perform a calculation such as for a wall loss signal.

As the program now stands all the programming has to be done in LISP. However, the program was written and the software chosen (ie it has a direct interface to C) so that it could also be done in C. To do this conversion is simply a matter of designing a new rule format to fit C syntax, rewriting the 7 line procedure PROCESSOR (page F-56) so that it accepts a rule written in C in the new format and rewriting in C the 2 line procedures GET-VALUE and PUT-VALUE (page F-3) in C so that the predicates written in C can access the values in the TUBE DEFSTRUCT structure.

Realistically it would appear to be difficult to design the system so the anyone unsophisticated in computing can add new rules. It will always come down to the fact that the rules depend on some features of the signal. If a new rule involved only the set of features already evaluated and resident in the TUBE DEFSTRUCT structure then there is no problem but if a new feature is required then it must be extracted numerically from the X - Y data. A general procedure to do this and allow simple descriptions of the desired features appears difficult and is probably not worth pursuing in this way. On the other hand there is a method (neural networks) that allows a user to "teach" a program how to characterize a pattern (such as those discussed here) by showing the program the pattern and giving it a label. It is based on the way that the human brain and eye works. The application of this technique to the eddy current problem needs further investigation.

Considering the value of a mature system to CNS it is not unreasonable to have a software house spend a man month or so a year adding and testing new rules. This is probably cheaper than developing the system described in the previous paragraph.

The processing of the rules is explained in the first part of Section 4.4. All of the rules used in the present version of EXPECT are contained in Appendix G and the English version of the rules is in Appendix H.

2.4 How to Run EXPECT

The following steps are needed to initiate and run EXPECT:

1. Enter the GCLISP environment.

2. Issue one of the following two commands: (LD 'LOADER) or (LF 'LOAFER) where the first command loads interpreted files and the second loads compiled files.

3. Issue the command (AT 'TUBE) where TUBE is the name of a file containing ANDES information on one tube (such as those in the first column of Table 4.1).

4. The screen will show a display such as Figure 3.1 for tube T-1-11. EXPECT will now accept the commands (see page F-44 and F-45): p, a, g, s, t, <an integer equal to one of the event numbers shown on the screen> or <any other key>. These commends have the following meaning:

p: toggle the printer toggle between on and off (see the <an integer ...> command below).

a: print structured information on every event onto the printer. Appendix A is a result of issuing this command.

g: dump the screen contents on the printer. Figure 3.1 is a result of issuing this command.

s: toggle the size of the screen resolution between 1 and 4 printer dots per screen pixel. Figure 3.1 has 4 printer dots per screen pixel.

t: plot a broken X at the tip of each event. Figure 3.1 has such broken X's while those in Figure 4.2 do not.

<an integer equal ...>: at the bottom of the screen is plotted the signal expanded in terms of the basis functions, the out and back expansions separately and in a third figure, together. The right parts of Figures 3.22 and 3.23 are examples of these expansions. If the printer toggle (see the p command above) is on, then information (such as shown in Figures 3.24 and 3.25) for the indicated event is printed out.

<any other key>: the next screen full of events is drawn on the screen (tube T-13-39 in Appendix C has 29 events and 3 screens of event plots). If the last screen is being shown then the program ends.

2.5 Guide to Appendixes E - I

These Appendixes contain the programs that were developed in connection with this project. They contain programs that are needed to run EXPECT, auxiliary programs (such as those used to produce Figures 3.4 - 3.6) and a program that was developed and used at one point in

EXPECT but is not used in the final version because its complication is not needed. This last program is a forward chaining inference engine and its listing is included because it may be needed in future versions where a more complicated control structure is required.

The program listings are divided into Appendixes E, F, G, and I. Appendix H contains the output from the procedure RULE-WRITER (page F-57) and is an English translation of the rules in Appendix G. Within each appendix there is one or more files, whose name is given at the top of the page with either a date (the first page of the file) or a page number (subsequent pages). Each file contains one or more procedures. The guides below are organized first by Appendix and then (where appropriate) by file. Page numbers are given in brackets after file names.

2.5.1 Guide to Appendix E

Appendix E contains the files listed in the variable UTILITY-LIST (see the EXPTLIST.LSP file below). These files are utility files.

EXPTLIST.LSP (E-1) This file defines four variables each containing a list of files. UTILITY-LIST contains a list of the files that make up Appendix E (auxiliary to EXPECT); FILE-LIST contains a list of the files containing procedures that are part of EXPECT, the files in Appendix F; RULE-LIST contains a list of the files containing the rules used by EXPECT (Appendix G); and TUBE-LIST contains a list of the files containing information on tubes.

LOADER.LSP (E-2) This file loads all the utility files, procedure files and rule files needed to run EXPECT. The files are in uncompiled form. The lists of EXPTLIST.LSP are used.

LOAFER.LSP (E-3) The same as LOADER.LSP except the compiled versions of the procedures are loaded.

DO-ALL.LSP (E-4 to E-5) This file contains procedures for printing or compiling all the EXPECT files, and for analyzing and plotting all the tubes at once. The lists of EXPTLIST.LSP are used.

PLOTTUBE.LSP (E-6 to E-8) Plots the separate X and Y ANDES data on an HP ThinkJet printer. Examples are shown in Figures 4.5 to 4.8.

READ-22.LSP (E-9 to E-11) The original ANDES data on all the tubes prefixed with a T in Table 4.1 were supplied one tube after another in one data file (on a disk with a bad bit). READ-22.LSP separates the data into one file per tube for all the tubes before the bad bit is encountered.

ADDER.LSP (E-12 to E-14) The procedures in this file were used to create Figures 3.4, 3.5, 3.6 and 3.19.

2.5.2 Guide to Appendix F

This Appendix contains procedures that are part of EXPECT itself. The variable FILE-LIST (see the file EXPTLIST.LSP in Section 2.5.1) contains a list of the files in this section.

HALOINIT.LSP (F-1) This file loads the files and sets the variables necessary to run the HALO graphics package.

ANALYZE.LSP (F-2) This file contains AT and ANALYZE-TUBE, the two procedures that can be used to initiate the analysis of a tube - see Section 2.4.

STRUCT.LSP (F-3 to F-14) This file contains all the procedures necessary for defining and manipulating the data structure that contains the tube information. It contains the GET-VALUE and PUT-VALUE macros (see Section 2.2.2) and all the DEFSTRUCT templates.

TUBE-GET.LSP (F-15 to F-17) TUBE-GET.LSP contains the procedures necessary for placing a filled TUBE data structure in RAM.

PARAMTRS.LSP (F-18) This file contains PARAMETERS-FILL, the procedure that fills the PARAMETERS slot in the TUBE defstruct structure (see Table 2.1).

INFO.LSP (F-19 to F-20) This file contains INFO-FILL, the procedure that fills the INFO slot in TUBE (see Table 2.1).

EVENT.LSP (F-21 to F-26) This file contains EVENT-FILL, the procedure that fills the EVENT slot in TUBE (see Table 2.1).

LIMITS.LSP (F-27) This file contains LIMITS-FILL, the procedure that fills the LIMITS slot in TUBE (see Table 2.1).

CALCPROP.LSP (F-28 to F-32) This file contains CALCULATED-PROPERTIES-FILL, the procedure that fills the calculated-properties slot in TUBE (see Table 2.1).

NOSP.LSP (F-33 to F-37) This file contains NOSP-FILL, the procedure that fills the NOSP slot in TUBE (see Table 2.1).

NON-NOSP.LSP (F-38 to F-41) This file contains NON-NOSP-Fill, the procedure that fills the NON-NOSP slot in TUBE (see Table 2.1).

ANALYSIS.LSP (F-42) This file contains ANALYSIS-FILL, an empty procedure (see Table 2.1).

DISPLAY.LSP (F-43 to F-55) This file contains the procedures that, given a filled TUBE structure, produce the displays such as Figure 3.1 (see Section 2.4).

RULE.LSP (F-56 to F-58) The RULE defstruct template, the procedure PROCESSOR (which evaluates the rules) and RULE-WRITER, the procedure that produces Appendix H (the English version of the rules), given Appendix G (the rules), are all in this file.

2.5.3 Guide to Appendix G

Appendix G contains the rules that are used in the present version of EXPECT. The variable RULE-LIST (see the file EXPTLIST.LSP in Section 2.5.1) contains a list of the files in this section.

PROSP-ID.LSP (G-1 to G-2) This file contains the rule for identifying prospective normal support plates (see Section 3.4.1).

NOSP-ID.LSP (G-3 to G-4) This file contains the rule for identifying normal support plate signals (see Section 3.4.1).

OD-ID.LSP (G-5 to G-6) This file contains the rule for identifying outer diameter wall loss (see Section 3.4.4).

ID-ID.LSP (G-7) This file contains the rule for identifying inner diameter wall loss (see Section 3.4.4).

HOLE-ID.LSP (G-8) This file contains the rule for identifying holes (see Section 3.4.4).

DENT-ID.LSP (G-9 to G-10) This file contains the rule for identifying mechanical deformation (see Section 3.4.6).

WHAL-ID.LSP (G-11) This file contains the rule for identifying Whalies (see Section 3.4.5).

WLSPC-ID.LSP (G-12 to G-13) This file contains the rule for identifying wall loss at a support plate center (see Section 3.4.12).

SCALL-ID.LSP (G-14 to G-15) This file contains the rule for identifying scallop bars (see Section 3.4.1).

SAT-ID.LSP (G-16) This file contains the rule for identifying saturated signals (see Section 3.4.22).

HAFSP-ID.LSP (G-17 to G-18) This file contains the rule for identifying half of a normal support plate (see Section 3.4.21).

SFRET-ID.LSP (G-19 to G-20) This file contains the rule for identifying a symmetric fret at a support plate (see Section 3.4.13).

2.5.4 Guide to Appendix H

This appendix contains the output of procedure RULE-WRITER, an English translation of the rules in the files of Appendix G (see Section 3.4).

2.5.5 Guide to Appendix I

This appendix contains the contents of file INF-ENG.LSP and is a forward chaining inference engine used in an early version of EXPECT. It was discarded in favour of RULE-PROCESSOR (page F-56) since the latter is much simpler and does all that is required (in this version of EXPECT). Most of the contents of INF-ENG.LSP come from LISP (2nd Edition) by Winston and Horn (Addison-Wesley, 1984) chapters 17 and 18. Future versions of EXPECT may require more complicated control structures and use this code.

2.6 Hardware and Software

The software used is Gold Hill's Golden Common LISP 286 Developer Version 2.2 with Halo Graphics. This was run on a Compaq Portable 286 with a 3 Mb Advantage! board and a 5Mhz 80287 numeric coprocessor.

GCLISP was an excellent development language for a project such as this one where one of the objectives was to quickly develop a prototype expert system. The combination of editor, interpreter and compiler provided an environment that allowed rapid code development. There were two complete overhauls of EXPECT during the course of this project and they were carried out rapidly with little difficulty. There was only one bug found in the product and this could be remedied with a simple fix.

The greatest weakness in GCLISP is the lack of a full set of numerical capabilities. For example, there is no double precision arithmetic and no two dimensional arrays. This weakness is compensated by GCLISP's direct interface to Lattice and Microsoft C.

GCLISP has the following features, related products and planned future features indicating that it is an industrial strength language suitable for this project:

1. Direct interface to Lattice or Microsoft C (ie C programs can call GCLISP procedures and vice versa). Since ANDES is written in C, it will allow the two programs to be easily integrated. This also means that the number crunching aspects of EXPECT can be translated into C where they will ran faster.

2. Gold Hill sells a product that allows the compiled program to run independent of the interpreter.

3. Should it be needed, Gold Hill sells an expert system shell embedded in GCLISP that reviews say is almost as powerful as LISP machine based shells such as KEE and ART. This product also has an interface to LOTUS 1-2-3, dBASE III and C.

4. Should it be needed, Gold Hill sells a 80386 based plug-in board that runs GCLISP about 6 - 8 times faster than the hardware/software used for this report. As well, there is a Compaq 386 version that runs about 3 times faster than the present version.

5. A new version of GCLISP is coming out in late 1987 that will be a full Common LISP (rather than the subset it is now) and will support the 80387 numeric coprocessor.

3. A CATALOGUE OF EDDY CURRENT SIGNAL INTERPRETATION EXPERTISE

3.1 Overview

This chapter catalogues the information on eddy current signal interpretation gathered during this project. As mentioned in Chapter 1, the problem has been limited to one type of probe (absolute) and one type of material (Incoloy 800 moderator heat exchanger - 1/2" tubing).

The full extent of this problem involves fifteen to twenty types of material (six common types), different wall thicknesses, different test frequencies and the two different types of probe (absolute and relative). It is estimated that the total amount of information would be at least four or five times that for the situation discussed in this report.

There are two types of signals that the analyst examines: normal signals that are not an indication of any problems in the tube and fault signals that do indicate a potential problem. As an example, Figure 3.1 shows the signals from a Moderator Heat Exchanger.

An important property of a fault is the degree of wall loss that a signal represents. Wall loss is the fault's maximum depth through the tube wall, relative to the tube thickness (for example 30% for Event 2 in Figure 3.1) so that it is an indication of how close the tube is to failure. If the wall loss is above some threshold depth, then the tube must be plugged (blocked off and not used again). An important criteria for determination of the threshold is the length of time to the next inspection (the length of time the tube must last).

The degree of wall loss is determined by the angle of the signal. For simple wall loss where the signal going out to the tip and returning to the balance point are the same (Event 2 in Figure 3.1), this angle is measured from the balance point (marked by a cross in Figure 3.1) to the tip (marked by a broken X in Figure 3.1) with respect to the - X axis and related to the calibration as shown in Figure 3.2. The same graph of wall loss vs. angle is used in more complicated situations but it is often difficult to determine where to measure the angle. Figure 3.3 shows an example of this.

Analysis of the signal is made difficult by the fact that there can be more than one defect at any given site or there can be a defect on top of a normal signal. There also can be signals (Whalies - see Section 3.4.5) that look almost like holes (to the untrained analyst), and are not defects but geometry variations in the tube.

Section 3.2 gives a few comments based on the interviews with the experts that were done during this project and how interviews with the experts would fit in the overall process of creating a mature EXPECT.

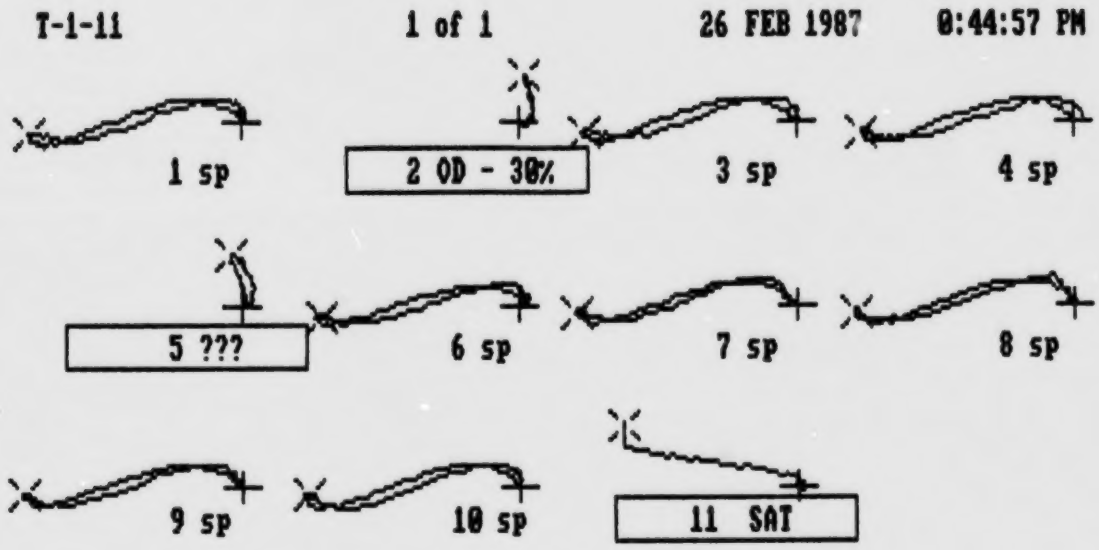


Figure 3.1 X-Y signals for a Moderator heat exchanger Incoloy 800 1/2" tubing. The events labelled sp are normal support plate signals, the event labelled OD - 30% indicates an outer diameter fault penetrating 30% of the way through the wall (30% wall loss), the event labelled ??? indicates that the program could not analyze the fault and SAT indicates that the signal is saturated. The cross indicates the inter-event zero point (balance point) and the broken X marks the tip of the signal as determined by EXPECT. The tip is important because the angle of the line between between it and the balance point determines the amount of wall loss (see Figure 3.2).

There are two approaches to analyzing a signal:

1. The recognize-and-analyze approach: simple recognition of the signal as belonging to a certain type by some of its features. The signal is then analyzed algorithmically. The isolated defect or isolated normal signal and some of the composite signals (more than one defect, or a normal signal and one or more defects occurring at the same spot) belong to this class.

2. The guess-and-test approach: clues are taken from the signal, a guess is made as to the cause and the guess is tested using models. This procedure is continued until a guess that predicts the signal is found. This is the method used in the more complicated composite signals. Section 3.3 is a general discussion of the above two approaches.

Figure 1 - Phase Angle* Versus Wall Loss

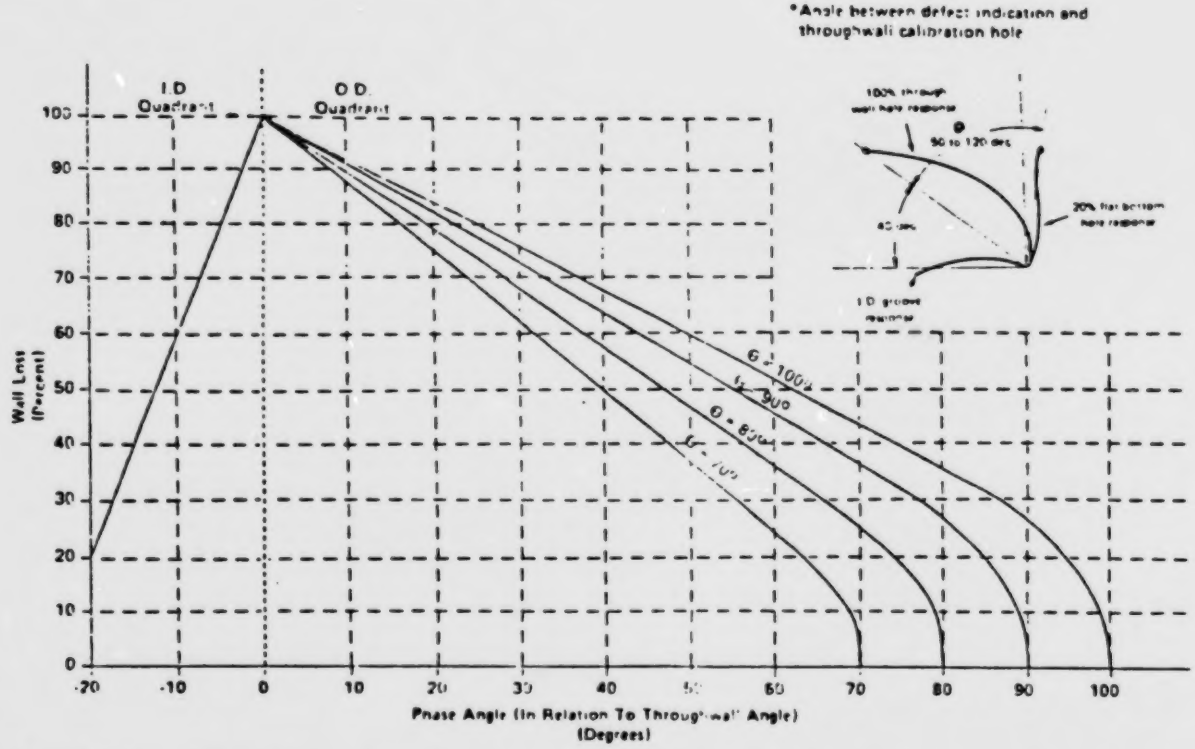


Figure 3.2 **Phase angle vs. wall loss.** As indicated in the insert, the X and Y gains are adjusted so that in a calibration tube a through hole makes a 40° angle with the -X axis, the ID groove is along the -X axis and the 20% flat bottom hole is 50° to 120° clockwise from the through hole. Then, given the phase angle and 20% flat bottom hole angle the wall loss can be determined. (From: E.E. Carter and A.P. Owen, Eddy Current Inspection Procedure for Steam Generator Tubes and Preheater Tubes, CNS report N-03640.1.07, 1985).

Figure 3.3 An example of a difficult to analyze signal along with some support plate signals.

Section 3.4 gives an discussion of the specific types of signals that can be expected in a tube, their causes, the rules for analyzing the signals and a description of their corresponding X-Y signals.

Section 3.5 describes how the EXPECT variables and properties referred to in Section 3.4 are calculated from the raw signal data.

3.2 Interviews with Eddy Current Experts

The information in this Chapter was gathered through separate one day conversations with two experts (NC Van Den Brekel and W Brouwer) and a half day session with their Supervisor (AP Owen), all of Central Nuclear Services (CNS).

The first interview (with AP Owen) was the most difficult in terms of extracting concrete rules because the interviewer only had a vague idea of how to interpret the signals and the interviewee was not really sure what the interviewer wanted. The domain for this project was narrowed down to that described at the beginning of this Chapter and the breadth of the total problem was explored. The two subsequent meetings were easier and quite productive.

Although there was much overlap, it was apparent that it was very useful to talk to more than one expert since they have different experiences and were able to present different information. It was also apparent that quite a few meetings would be required to cover all the topics.

There is some information that the program needs that cannot be supplied by the experts during an interview. Statistical data such as the normal angular or length variation of support plate signals in a single tube or the variations from tube to tube or the range of variation of "shape" that a particular defect can have. The analysts judge these things by eye and use their experience but the computer has to have numbers with which to work. The sit-down-and-talk type of interview is not adequate to convey all the information that the computer program needs. This information can only be gathered by testing the rules on a large number of tubes.

Based on these experiences the following steps will be needed to develop a working system:

1. Have initial interviews - preferably with all available experts
2. Based on 1. develop and program an initial set of rules
3. Obtain an initial set of tube data
4. Test the rules with all available tube data
5. Go back to the experts and discuss the results
6. Make changes in the rules and add some new rules
7. Obtain new tube data
8. Go to 4.

This process should converge after a few iterations, except for new interpretations (about four per year for the whole eddy current testing domain). It is important to add these new interpretations to EXPECT as they come along because without them EXPECT would rapidly become obsolete. For this reason a significant part of the present investigation addressed the issue of adding new rules independent of EXPECT's author. (See Chapter 2 for a discussion of this).

Note the continual acquisition of new tube data. It is quite important to keep testing the rules with new data. This is the best way to discover the limitations of the program.

In practice, a lot of effort is involved to ensure that all the cases are covered and all the rules are tested properly. The iterative process described above is designed to accomplish this.

EXPECT is presently on step 4, having progressed from step 3.

3.3 Event Analysis

This Section is divided into two parts:

1. A discussion of the analysis of events due to an isolated defect or isolated normal signal.

2. A discussion of events due to a normal signal and one or more defect signals or two or more defect signals occurring at the same spot (composite signals).

3.3.1 Events Due to Single Causes

Some signals, such as an isolated defect or a normal support plate signal, are easily recognized and analyzed (the recognize-and-analyze type of signal mentioned in Section 3.1). For example, to recognize a normal support plate signal such as those marked "sp" in Figure 3.1, the analyst looks for the following characteristics:

1. The angle from the balance point to the tip is near the - X axis.

2. The initial angle at the balance point is in the 1st or 2nd quadrant.

3. The signal going out from the balance point to the tip is approximately the same as the signal returning from the tip to the balance point.

4. The signal has an "S" shape (this is quantified in Section 3.5).

5. The line from the balance point to the tip is similar in length and angle to the others in the same tube with the first four characteristics.

Another example is an isolated outer diameter wall loss (Event 2 in Figure 3.1):

1. The event is not a normal support plate signal.

2. The signal going out from the balance point to the tip curves towards the - X axis ie counter-clockwise.

3. The angle from the balance point to the tip is between 44 and 110 degrees (the numbers 44 and 110 depend on the calibration) with respect to the - X axis.

4. The signal going out from the balance point to the tip is approximately the same as the signal returning from the tip to the balance point.

Both of the above types of signals are distinctive and the analyst could, by eye, recognize the events in Figure 3.1 (and similar types of events) within 1 to 2 seconds or thereabouts. Determining the depth of a defect such as Event 2 is then simply a matter of measuring the angle to tip and using a graph such as Figure 3.2. The analyst can usually make this estimate of the depth and record the result within about 30 seconds. This is a figure of merit for which EXPECT should aim.

In addition to angle, the other features that an analyst examines are magnitude (length from

balance point to tip) and the width of the signal on the strip charts (ie the length of the signal in time). The former is a measure of the volume of material removed and the latter is a measure of the axial length.

To analyze the recognize-and-analyze type of signal by computer is a matter of determining those features that the analyst judges by eye characterize a particular type of signal, choosing the best algorithms to pick out these features (such as the "S" shape), determining the normal variation of the parameters (such as the angle that a normal support plate signal makes with respect to the -X axis) and testing the resulting rules. Section 3.4 gives detailed examples of this process and as can be seen EXPECT is very successful with this type of signal.

3.3.2 Composite Signals

When more than one defect occurs at (or near) the same spot or a defect occurs at the same spot as a feature producing a normal signal, then the analysis can be more complicated.

For some of these composite cases it is, as for the isolated signals, simply a matter of measuring an angle (with experience and past analysis showing where to make the measurement). An example of this is a symmetric fret (or wall loss) at a support plate center - see Section 3.4. These cases can be implemented on the computer as described above.

In the more difficult cases (unusual or new situations) a guess-and-test type of approach rather than the algorithmic approach is used. The analyst takes some clues from the signal and makes a guess at the causes, tests the guess by comparing the unanalyzed signal with the signal made up of the addition of models for the individual defects or normal signals that are hypothesized to make up the composite signal, adjusts the guess and so on until the guess fits the signal. This last procedure is somewhat of an an art and takes considerable experience to do properly. The models used are the typical X and Y strip chart traces of the normal and defect signals. It seems that the analyst does this process largely in his head, using experience as a guide.

To illustrate the difficulties involved consider Figures 3.4 - 3.6. Each of these figures was created by vectorialy adding two signals from Figure 3.1, as indicated in the caption. The X strip chart signals of each of the composite signals (ie those of Figure 3.4 - 3.6) is the sum of the two strip chart X signals of the two components (event 1 and event 2 of Figure 3.1 for Figure 3.4). The Y signal of the composites was constructed similarly. This produces realistic results for the situation described in this report (an example where it does not is when the support plate is nonferromagnetic and the interaction between the support plate signal and the defect signal is nonlinear). The 11 different signals represent differing relative displacements (with respect to each other) of the beginnings of the two components. The signal labelled 1 represents the defect at the leading edge of the support plate signal; 2 through 5 are the result as the defect is moved towards the center of the support plate signal; in 6 the defect is at the support plate center; and 7 through 11 show the result of moving the defect signal through to the other side of the support plate signal.

For some of the composite signals (1, 2, 6, 10 and 11 in Figure 3.4 and 3.6) it is relatively easy to recognize the two components and even to make measurements to extract the necessary information (ie the angle of the defect signal), but for signals such as 4 or 9 in all three of the figures, it is somewhat more difficult. An added complication is that normal support plate

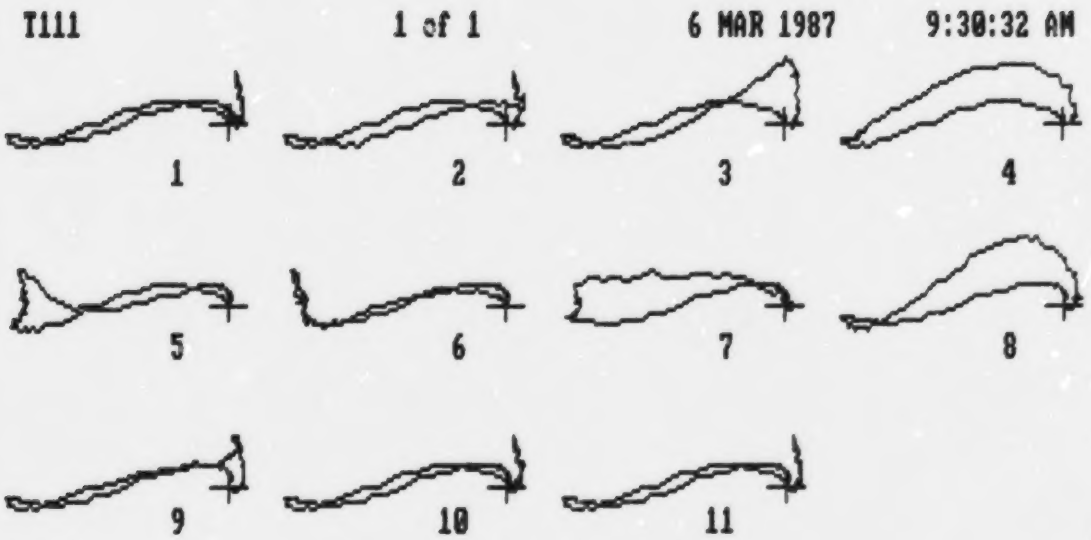


Figure 3.4 The addition of a normal support plate signal and a defect signal. The support plate signal is event 1 of Figure 3.1 and the defect signal is event 2 of Figure 3.1. The different figures represent different relative displacements of the two signals (see text).

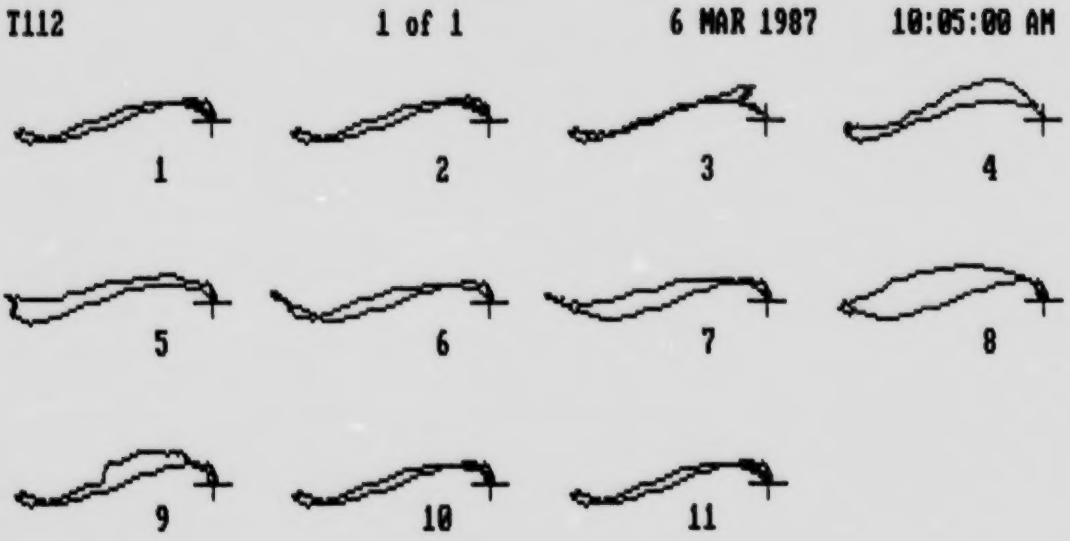


Figure 3.5 The addition of a normal support plate signal and a defect signal. The support plate signal is event 1 of Figure 3.1 and the defect signal due to a hole (from another tube). The different figures represent different relative displacements of the two signals (see text). The hole signal is quite small and since its angle is the same as the initial angle of the support plate, it is difficult to see in 1, 2, 10 and 11.

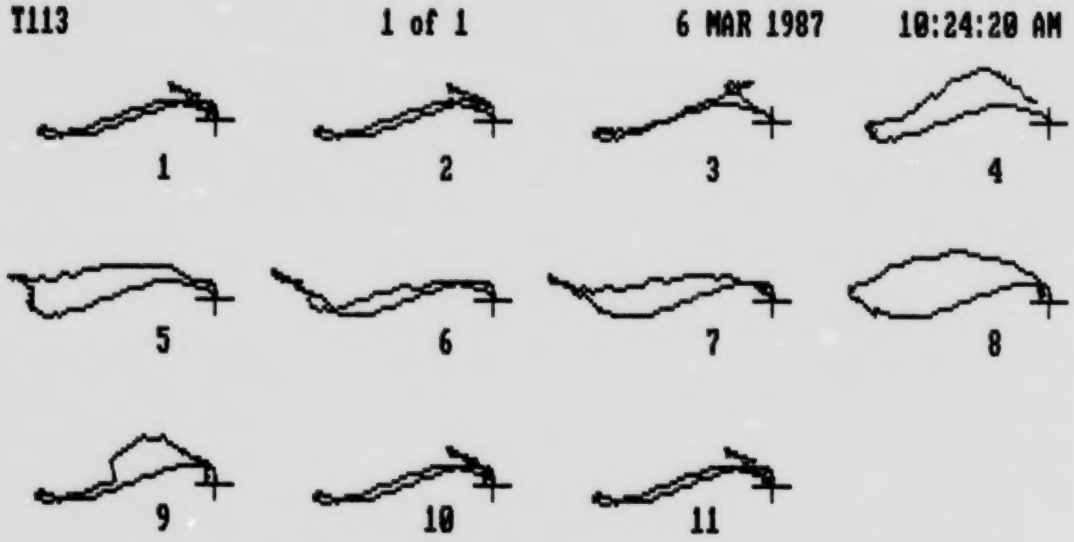


Figure 3.6 The same as Figure 3.5 except that the hole signal of Figure 3.5 has been doubled.

signals in different tubes have slightly different shapes. This makes the subtle differences among the 4th composite in the three figures that much more difficult to see. In Figures 3.5 and 3.6 events 5 and 7 in the same Figure look like different defects even though they are the same. These differences are important: the events in Figure 3.5 and Figure 3.6 would cause the tube to be plugged while those in Figure 3.4 would not.

There appears to be several ways to analyze the composite type of signal on the computer:

1. Use the rule-based recognize-and-analyze approach of the previous Section.

2. Provide the analyst with the data base and an vector adder (or even a vector subtracter for those signals that add linearly) so he can (on-line) try to reproduce the problem signal.

3. Do a full multi-parameter least squares fit to the signal using a data base of defect and normal signals and a method of adding the isolated defect signals to produce a composite.

4. The analysis is done in two steps: the first step to find an approximate decomposition is done by the analyst (as in 2) and the second step to find a detailed decomposition is done by a least squares fitting program.

5. The same as 4, except the first step is done using a rules based approach.

A potential difficulty with all of these approaches is the uniqueness of the decomposition of the composite signal. This issue and the five approaches are discussed below.

3.3.2.1 Recognize-and-analyze Approach to Decomposing the Composite Signals

The first step in a rigorous (one that ignores the issue of uniqueness in decomposition) recognize-and-analyze method of analyzing composite signals would be to enumerate the set of isolated signals. Secondly, identify those signals that could exist in pairs (for example [support plate + hole] is possible while [support plate + tube sheet] is not), triples etc. Then for each legal pair, triple etc., generate a set of rules.

That this will be difficult can be seen by examining just one case: support plate plus wall loss. The variables that must be considered are:

1. the angle of the defect signal

2. the amplitude of the defect signal

3. the relative position of the defect and support plate

4. the number of different types of support plate signals.

Each of the first three variables will have to be quantized with the following being reasonable estimates for the number of different values needed for an accurate representation: angle (15); length (10); relative position (20) (finer gradations than shown in Figures 3.4 - 3.6 would seem to be needed). This means 3000 rules for each type of support plate. The situation shown in

Figure 3.19 is even worse since there are 5 variables (2 angles, 2 lengths and a relative position) to consider. Since this approach will lead to too many rules (by orders of magnitude), in theory it clearly is not appropriate.

However, in practice the (probable) nonuniqueness of the decomposition will lead to far fewer rules than this and will result in an analysis which includes a range of values. More work needs to be done in this area.

As part of the effort in exploring the above it is necessary to have a way of vectorialy adding isolated defect signals to produce composite signals. The reason is that in order to be able to develop the rules as described above it is necessary to have examples of all the many ways in which two three or more signals can be combined and it is not realistic to expect to be able to find all those combined signals from real tube data. Firstly, it is unlikely that every case will be there and even if they are it may not be found or, if found, it may not be digitized (ie be pre-ANDES). It is much better to be able to generate composite signals at will. For cases where the signals can be added linearly (as in Figures 3.4 - 3.6 and 3.19) this is trivial, but in cases where the signals interact nonlinearly, it may not be straightforward or even possible. This latter case has not yet been investigated.

A vector subtracter might also be useful for those cases where one of the signals is known (such as the support plate signal plus wall loss) and the signals combine linearly. In other cases, such as the sum of two wall loss signals, where neither signal is known, it could not be used.

In addition to the adder it will be necessary to have a data base of isolated signals for it to add or a way of generating the isolated signals. Simple models (rather than real data) to generate, for example, isolated wall loss at any angle or length, should be sufficient since the result will be used to build rules, which, experience has shown, can be in terms of relatively crude representations of a signal's features. Either real data or models for the support plate signals (as appropriate) can be used.

If it turns out that a lot of rules are needed, then it will be necessary to have a more sophisticated control structure than is used in the present version of EXPECT.

3.3.2.2 Analyst with Adder Approach to Decomposing the Composite Signals

Another approach, much easier to implement, would be to use the same database of isolated defect and normal signals along with the same adder discussed above, as tools for the analysts to test their guesses at the decomposition of a problem signal. This could be done on-line so that, for example, if an analyst wished to analyze an unknown signal that looked like 4 in Figure 3.6 he could call up a normal support plate signal from the tube being analyzed, add to it (from the database) a first guess at the defect signal. If the first guess was OD - 30% then a signal like 1 in Figure 3.4 would be produced. Using the arrow keys (or pointing device), the defect signal could be continuously moved through the support plate signal until the best match to the unknown signal were produced, which would be like 4 of Figure 3.4. Leaving the relative positions of the components the same, the angle and length of the defect signal could then be independently and continuously varied on-line until the best were obtained. This would be like signal 4 of Figure 3.6. Or, in a semi-automated mode, one of these two variables (the analysts choice) could be varied and the other, in real time, could be adjusted to give the best fit. The operator would then stop varying the parameter when he judged, by eye, that the best overall fit had been found.

Alternatively, it may be better to subtract the support plate signal and use the recognize-and-analyze method for isolated signals on the difference. But of course the problem of finding the relative positions of the two signals still remains. Which of these two approaches to use would be at the analysts option and depend on the situation. Figure 3.19 shows a composite signal made up of two defects. In this case, the subtraction approach could not be used because neither of the component signals is known and the analyst would not know what to subtract.

3.3.2.3 Least Squares Fit Approach to Decomposing the Composite Signals

With the adder/data-base tools mentioned above, another obvious approach to consider is a least squares fit of the models to the problem signal. However, given the (probable) nonuniqueness of the composite signals and the number of variables involved (three for support plate + wall loss, five for two wall loss signals) it would appear that the function to be minimized would be plagued with many local minima and no strong global minimum. This would make this approach, by itself, quite difficult.

The degree of reliability of the fitting procedure will depend on the number of parameters. As the number of parameters increases, the uniqueness decreases and the difficulty of finding a strong solution increases. Without having tried, it looks possible to automate when three parameters are involved but with five parameters it looks very difficult. In any case, this cannot be determined without developing a prototype and testing it thoroughly.

3.3.2.4 Analyst Plus Least Squares Fit Approach to Decomposing the Composite Signals

This is a variation on the approach of 3.3.2.2 in which the analyst uses the adder and data base (and experience) to get an approximation to the decomposition and then a least squares fitting program takes over at the end to find the best decomposition. In practice, this will work only when the analyst can get close to a minimum. When one variable is involved, this procedure is the same as the semiautomatic mode mentioned in 3.3.2.2.

3.3.2.5 Recognize-and-analyze Plus Least Squares Fit Approach to Decomposing the Composite Signals

This is the same as 3.3.2.4 except the analyst is replaced by a set of rules. A final version of EXPECT would probably be based on this approach.

3.3.2.6 The Problem with Non-uniqueness in the Decomposition

As mentioned several times above, there appears to be a problem with uniqueness in decomposition of the composite signals. Comparing events 4 of Figures 3.4 to 3.6 (they look almost the same) or examining events 5,6 and 8 of Figure 3.19 (there seems to be no obvious clues that indicate a hole signal) supports but does not prove this contention. More work needs to be done in this area.

In terms of interpretation of composite signals this will likely result in a range of decompositions rather than a single value. The range will depend on the signal. For example, a signal with few distinct features, such as event 8 of Figure 3.19, might have a wide range of decompositions whereas one like event 3 of Figure 3.19 would probably have a unique decomposition, since it has two very distinguishing features.

The determination of the error bars on an analysis will probably be done using a set of rules-of-thumb based on experience with the adder and data base of isolated signals.

The common need of all of the above approaches is the development of the adder and the data base of isolated signals so this should be the next step. This would allow the implementation of the approach of 3.3.2.2 and it may turn out that this successfully addresses all of CNS's needs for this system and further development is unnecessary. These tools will also be necessary in a non-uniqueness study.

3.4 Types of Eddy Current Signals and Rules of Analysis

Table 3.1 summarizes the information given in this section. It also lists the status of implementation of the rules of analysis in the present version of EXPECT for each type of signal.

The signals are categorized in Table 3.1 as follows:

1. Normal Signals. These signals are expected in every tube and are caused by normal features in the tube or support structure.

2. Isolated Defect Signals: These signals are not expected and may or may not be present. Most of them can be serious enough to cause a tube to be plugged, although some are not.

3. Composite Signals. These signals are composites made up of the sum of two or more signals in the same location. This includes signals made up of two or more defects and those made up of a normal signal plus one or more defects.

4. Other Signals.

This section documents the rules implemented in EXPECT. There are 12 rules covering (fully, partially or as a special case) 40% of the different types of signal. For 60% of those cases where rules were not implemented, there is a description of a rule that could have been implemented (were there time). For the rest of the cases, not enough information was obtained to describe a rule.

The rest of this section is organized as follows. Each subsection (by number) corresponds to a type of eddy current signal listed in Table 3.1. For example, subsection 3.4.5 discusses Whalies. The first part of each section is taken up by a description of the signal and the second part by analysis techniques (if any).

Appendix H contains a list of all the rules used in the present version of EXPECT. It is a printout of file RULE-ENG.LSP and is produced by the procedure RULE-WRITER of file RULE.LSP. RULE-WRITER gets its information directly from the same procedures that EXPECT uses, so that its listing is always consistent with EXPECT. The Rule numbers do not follow the numbers in the Table.

If a rule is implemented in EXPECT then it is discussed in terms of the predicates in the IF part of the rule. The rules are reproduced as Figures in this Chapter. The discussions of unimplemented rules will be as detailed as possible and will reference the same predicates and methods where appropriate, but will, in general, not be fully quantified.

To determine whether an event obeys a rule, EXPECT tests each component of the IF part of the rule against the properties of the event. If all the components are true then the THEN part of the rule is evaluated. Evaluation of the THEN part entails recording the analysis for that event and, if necessary, calculating the wall loss or other parameter. The return value of this procedure appears under the event in the display as EXPECT'S analysis. Examples are "sp" or "OD - 30%" in Figure 3.1.

The predicates described in each subsection are in the order in which they are given in the corresponding rule. Also listed are the THEN procedures and their return values. The words in capital letters in the descriptions of the predicates refer to EXPECT LISP variables. They are described in in each subsection and quantified in Section 3.5.

In general a variable is a list over the events in a tube. For the purposes of the rest of this Chapter, we will pretend that the variables are single valued, since it will be much easier to say "The TIP-ANGLE is ..." rather than having to continually repeat "The i^{th} value of the list TIP-ANGLE is ... ". It will be understood that when the variable is referred to, a particular event is meant. The predicates return either T (true) or nil (false in LISP). For each predicate a reference is given to the page in Appendix G containing the listing.

3.4.1 Support Plate

A sketch of a heat exchanger tube and some of the structures that produce normal signals are shown in Figure 3.7. A support plate is a piece of metal which keeps the tube from sagging. Figure 3.1 shows some normal support plate signals as does Figure 3.8. There are various types of support plate signals each of which give different signatures: full (such as in Figures 3.1 and 3.8) or half (only one half of the plate present - see Figure 3.9); baffle plate; clamp bar or stay rod . The support plate in the straight section is called a baffle plate. A support plate in the u-bend section is called a clamp bar or scallop bar. The tube in this region has an oval cross section and the clamp bar has a different signature from the baffle plate. In some of the older moderators, the vibration problems were bad enough that extra supports (stay rods) were necessary. Usual practice for the analysis of clamp bar signals is to compare them among themselves and the next tube, and if the signals are all the same, to assume that they are normal. An added factor is that the materials of the support plate may either be different from the tube material or the same as the tube material.

Table 3.1 List of Types of Eddy Current Signals. For some types of signals, insufficient information was obtained during the interviews to be able to develop an analysis technique. These signals have a "No" in column 3 of the Table.

Signal Name	Type of Signal	Analysis Technique Described in Section 3.4?	Technique Implemented in Expect?
1. Support plate	Normal	Support plate: Yes Clamp bar: Yes (but see text) Half support plate: Yes (but see text)	Yes Yes No
2. Tube sheet	Normal	Yes	No
3. U-bend drift	Normal	Yes	Yes (but see text)
4. Wall loss	Isolated defect	Yes	Yes
5. Whalies	Isolated defect	Yes	Partially (see text)
6. Mechanical deformation	Isolated defect	Yes	Yes
7. Deposit	Isolated defect	No	No
8. Resistivity variations	Isolated defect	No	No
9. Magnetic permeability changes	Isolated defect	No	No
10. Intergranular attack	Isolated defect	No (see text)	No
11. Support plate plus wall loss	Composite	Yes	Special case implemented
12. Fretting at a support plate	Composite	Yes	Special case implemented

Table 3.1 (Cont'd)

Signal Name	Type of Signal	Analysis Technique Described in Section 3.4?	Technique Implemented in Expect?
13. Tube sheet plus wall loss	Composite	Yes	No
14. Support plate plus mechanical deformation	Composite	Yes	No
15. Pit within a pit	Composite	Yes	No
16. Dent	Composite	Yes	No
17. Other combinations	Composite	Yes	No
18. Change in balance point	Other	Yes	See Text
19. Probe Wobble	Other	Yes	No
20. Characteristics of Andes	Other	Partially	Half of a normal support plate signal can be identified
21. Saturated Signal	Other	Yes	Partially
22. Die chatter	Other	No	No

The identification of support plate signals is a two pass procedure (see Figure 3.10). The first pass (Rule 1) looks at the angle between the balance point and the tip (Figure 3.1), the initial angle, that the signal going out to the tip is the same as the signal returning to the balance point and that the signal has an "S" shape. Any event that obeys Rule 1 is called a PROSP (prospective normal support plate) in the program. The second pass (Rule 2) averages the tip length and tip angle of those selected by Rule 1, and identifies those that are within a certain tolerance of the averages (10% and 5°, respectively, for the type of tubing discussed here). These events are labelled as NOSPs (normal support plates) in the program.

A two pass procedure is necessary for several reasons. There is significantly more variation in support plate tip angle and tip length between tubes than among signals in the same tube. The tip angle of a normal support plate signal could be anywhere within a 40° range but within a single tube all the normal support plate signals are usually within 5° of each other (actually all of the parameters, such as 10% and 5°, need more testing to verify). There are defect signals (symmetric fret - see 3.4.12 and Rule 12) that have the same shape and size as a support plate signal but differ in that they are rotated clockwise with respect to a normal support plate signal by an amount proportional to the amount of wall loss. Therefore, without knowing the average normal support plate angle in the tube being analyzed, the symmetric fret could not be separated from the normal support plate signal. Similarly, if there is a pit at the centre of the support plate (see event 6 in Figures 3.4 - 3.6), the major effect is to lengthen the signal. This is difficult to pick out without knowing the average tip length.

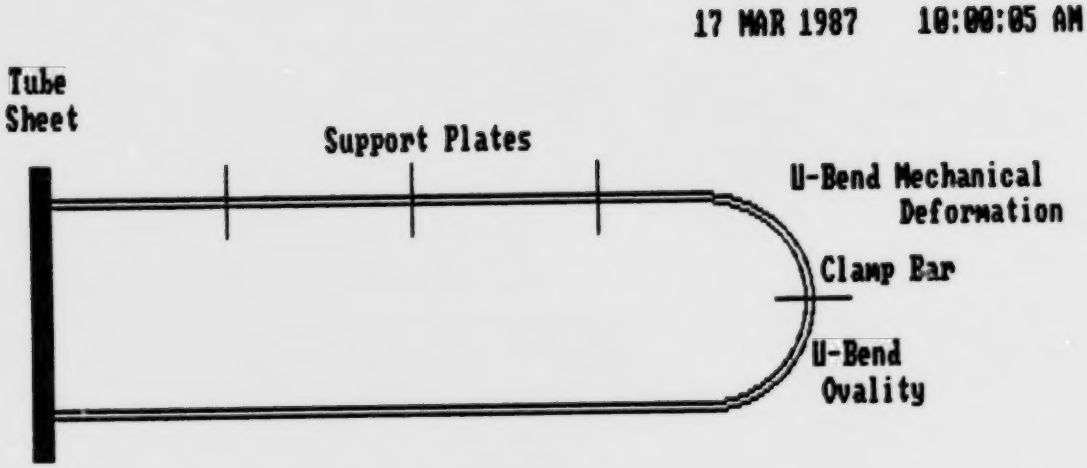
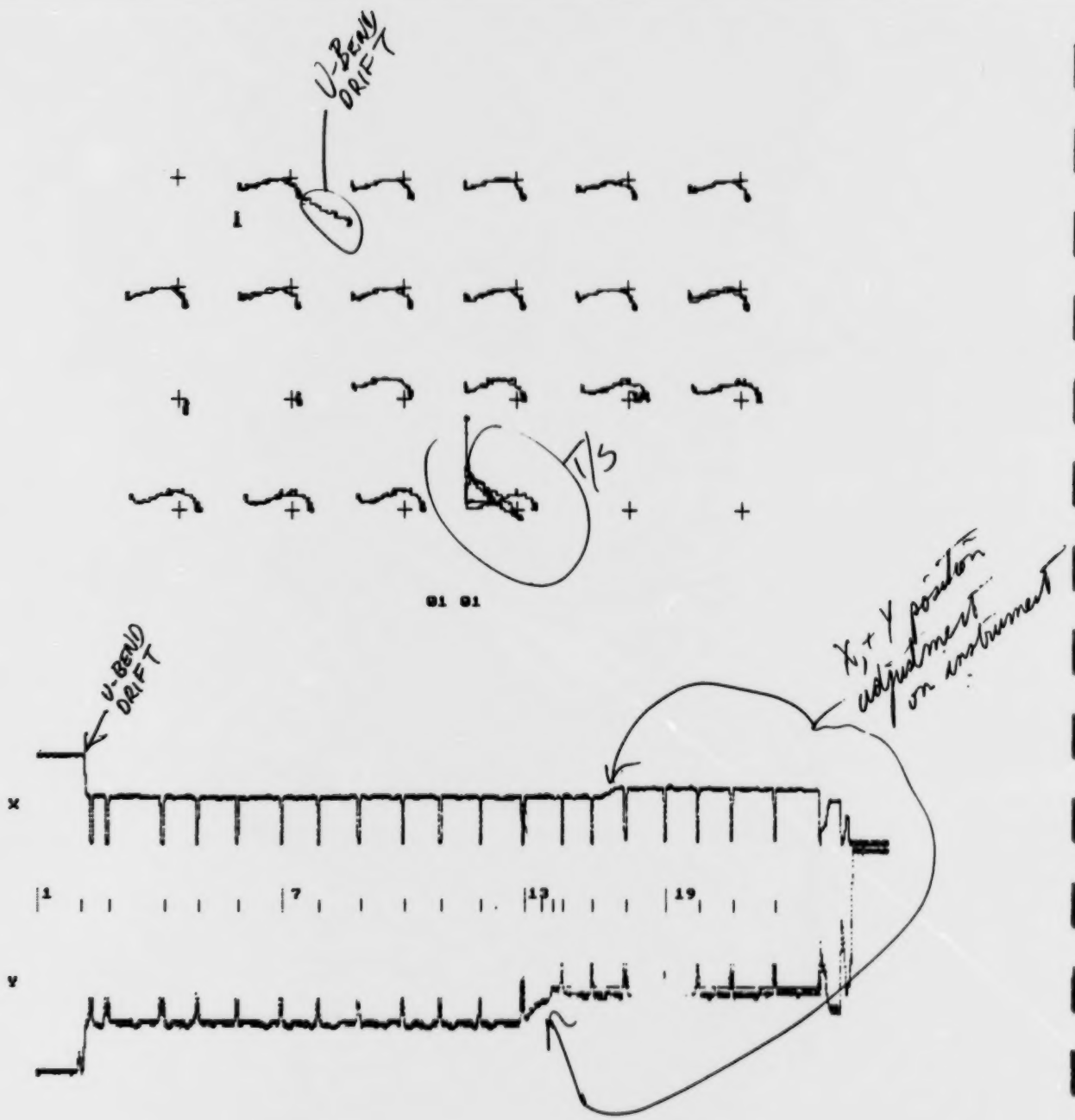


Figure 3.7 Sketch of a heat exchanger tube with causes of normal signals (not to scale). The radius of curvature in the u-bend section differs among tubes in the same heat exchanger. In those with the smaller radii of curvature there is sometimes mechanical deformation at the beginning of the bend and the tube cross section in the bend is sometimes oval.



PNGS	TUBE NUM.	Freq: 145
Unit 6		86-07-10
Moderator Hx	01 01	11:44
Hx N7058D		Supervisor: EIC

Page 1 of 1

Figure 3.8 **Some normal signals. The tube sheet signals (labelled by T/S) and the u-bend drift are indicated. The change in balance point (labelled as X + Y position adjustment ...) was done by the analyst. These signals were produced by the ANDES system.**

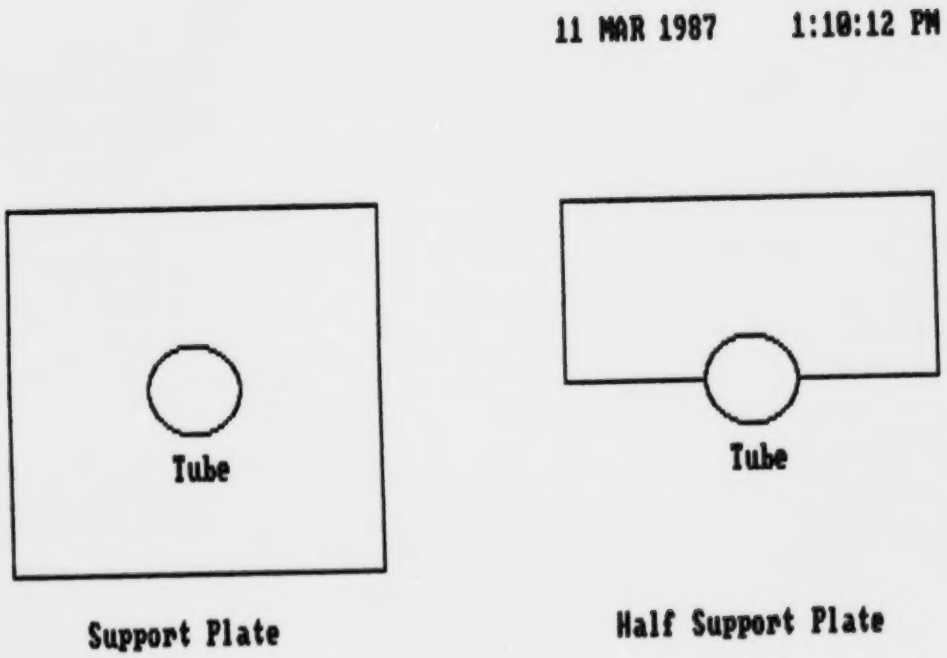


Figure 3.9 A sketch of a support plate and a half support plate.

```
RULE 1      [PROSP-ID.LSP]

 IF
      THE TIP ANGLE IS WITHIN + OR - 20 DEGREES OF THE -X AXIS
      [T-A20]
  AND THE INITIAL ANGLE (BOTH OUT AND BACK) IS IN THE 1ST OR 2ND QUADRANT
      [IA-Q]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B]
  AND THE S-SHAPE IS PRESENT IN BOTH THE OUT AND BACK SIGNALS
      [S-SHAPE]

 THEN
      THE EVENT IS PROVISIONALLY DUE TO A NORMAL SUPPORT PLATE (PROSP)
      [PROSP-ANALYSIS]

                          - - + - -

RULE 2      [NOSP-ID.LSP]

 IF
      THE EVENT IS A PROSP
      [PROSP-TEST]
  AND THE TIP LENGTH IS WITHIN nosp-length (FRACTION) OF THE PROSP
      AVERAGE TIP LENGTH
      [PROSP-AVG-LEN]
  AND THE TIP ANGLE IS WITHIN nosp-angle DEGREES OF THE PROSP AVERAGE
      TIP ANGLE
      [PROSP-AVG-ANGLE]
 THEN
      THE EVENT IS DUE TO A NORMAL SUPPORT PLATE (NOSP)
      [NOSP-ANALYSIS]

                          - - + - -
```

Figure 3.10 Rules 1 and 2 in EXPECT, used to detect a support plate signal. All of the events are tested against Rule 1 before any are tested against Rule 2. Any that pass Rule 1 are used in th e calculation of the PROSP averages of Rule 2. Each component of the IF part of the Rule has two pieces of information: an English translation of the property being tested followed by the name of the LISP predicate that EXPECT calls in square brackets at the end. The name in square brackets beside the rule number is the name of the LISP file containing the rule and the predicates.

PROCEDURES USED IN RULE 1 - PROSP (SUPPORT PLATE STEP 1)

T-A20: This predicate checks to see if the TIP-ANGLE is between 20° above the -X axis and 20° below the -X axis. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1 (page G-1).

IA-Q: True is returned if INITIAL-ANGLE-OUT and INITIAL-ANGLE-BACK are between 0 and 180° (with respect to the -X axis). INITIAL-ANGLE-OUT is the angle in degrees with respect to the -X axis of a line tangent at the balance point, of the signal going out from the balance point to the tip. INITIAL-ANGLE-BACK refers to the signal returning from the tip to the balance point (page G-1).

O=B: Returns true if OUT=BACK? is true. OUT=BACK? is true if the signal going out from the balance point to the tip is approximately the same as the signal coming back (page G-2).

S-SHAPE: If both S-SHAPE-OUT? and S-SHAPE-BACK? are true then true is returned. S-SHAPE-OUT? looks for the characteristic "S" shape of the support plate for the out half of the signal. The "S" shape is characterized by a maximum followed by an inflection point. An early version of the program also checked for a minimum after the inflection point, but it was found that some support plate signals did not have one. This algorithm still needs more testing. S-SHAPE-BACK? refers to the return half of the signal (page G-2).

PROSP-ANALYSIS: If called returns "PROSP", indicating that the event is a prospective support plate signal (page G-2).

PROCEDURES USED IN RULE 2 - NOSP (SUPPORT PLATE STEP 2)

PROSP-TEST: This predicate returns true if the event has been identified by rule 1 as a prospective support plate (page G-3).

PROSP-AVG-LEN: Returns true if the TIP-LENGTH is within NOSP-LENGTH of PROSP-AVG-TIP-LENGTH. TIP-LENGTH is the length of a line from the balance point to the tip. PROSP-AVG-TIP-LENGTH is the average TIP-LENGTH for all those events that were identified as PROSPs. All averaging is done before Rule 2 is processed (see file NOSP.LSP). NOSP-LENGTH is a fraction (of PROSP-AVG-TIP-LENGTH), with .1 being the value used for the results in this report (page G-3).

PROSP-AVG-ANGLE: True is returned if the TIP-ANGLE is within NOSP-ANGLE of PROSP-AVG-TIP-ANGLE. TIP-ANGLE is the angle with respect to the -X axis of a line from the balance point to the tip. PROSP-AVG-TIP-ANGLE is the average TIP-ANGLE for all those events that were identified as PROSPs. The value of NOSP-ANGLE used for the results in this report was 5° (page G-3).

NOSP-ANALYSIS: If called, returns "sp", indicating that the event is a support plate (page G-4).

```
RULE 9      [SCALL-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE TIP LENGTH IS 15% to 25% THAT OF THE AVERAGE NOSP TIP LENGTH
      [SCALL-SIZE?]
  AND THE TIP ANGLE IS -20 to -30 DEGREES LESS THAN THAT OF THE AVERAGE
      NOSP TIP ANGLE
      [SCALL-ANGLE?]
  AND THE SIGNAL CURVES COUNTER-CLOCKWISE
      [CURVE-CCW?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]
  AND THE INITIAL ANGLE IS IN THE SECOND QUADRANT
      [IN-QUAD-2?]

 THEN
      THE EVENT IS DUE TO A SCALLOP BAR
      [SCALLOP-BAR]
```

- - + - -

Figure 3.11 Rule used for the identification of clamp bars. Refer to the caption of Figure 3.10 for the interpretation of this rule.

Clamp bar signatures are a little different as shown in Figure 3.11, where they are called scallop bars.

PROCEDURES USED IN RULE 9 - SCALLOP BAR

NON-NOSP: This predicate is true if the event was not identified by Rule 2 as a NOSP (normal support plate). Note that it still could have been identified by Rule 1 as a PROSP (page G-5).

SCALL-SIZE?: The size of the scallop bar is much smaller than a baffle plate. This predicate returns true if the TIP-LENGTH is between 15 and 20% of NOSP-AVG-TIP-LENGTH. TIP-LENGTH is the length of a line from the balance point to the tip. NOSP-AVG-TIP-LENGTH is the average value of TIP-LENGTH over the events that passed Rule 2 (page G-14).

SCALL-ANGLE?: Returns true if TIP-ANGLE is 20° to 30° less than NOSP-AVG-TIP-ANGLE. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1. NOSP-AVG-TIP-ANGLE is the average value of TIP-ANGLE over the events that passed Rule 2 (page G-14).

CURVE-CCW? The same as CURVE-TO--X? - see Rule 3, Section 3.4.4.

O=B?: Refer to Rule 3, Section 3.4.4.

IN-QUAD-2?: Refer to Rule 7, Section 3.4.5.

SCALLOP-BAR: If called returns "sc bar" (page G-15).

3.4.2 Tube Sheet

The tube sheet, also shown in Figure 3.7, is the supporting metal at the ends of the tube. Figure 3.8 shows some tube sheet signals (marked T/S).

Since the tube sheet is usually a different material from the tube, the signal saturates and there is no return signal. It is only the first part of the signal that has any information. In general, the signal goes from the balance point along the -X axis and then saturates. Figure 3.8 shows a typical tube sheet signal (marked T/S on the X-Y plots and the rightmost set of 2 or 3 peaks on the strip chart). If the initial angle is pulled into the second quadrant then it is probably an indication of a pit at the tube sheet and if the signal makes a small excursion into the first quadrant then there is probably some mechanical deformation at the tube sheet. Since there is no return signal and only two signals per tube there is no way to compute meaningful averages as there is with support plates and this makes it much more difficult to analyze. Signals from different tubes in the same heat exchanger are usually very similar, and EXPECT could use this information in the same way it does now for support plate signals in the same tube. In any case, more work is required to properly parameterize tube sheet signals.

3.4.3 U-bend Drift

This is a drift in the balance point caused by the probe entering the bent part of the tube. This happens because the probe, which was centered in the straight part of the section tends to move off center as it moves around the bend. When the radius of curvature, which can vary from 2.5 inches to 2 feet, is too small (as in Figure 3.8) then the probe cannot get around the curve and the rest of the tube is tested by inserting the probe into the other end.

If the probe had been able to get around the bend in Figure 3.8, the clamp bar signals would have been similar to (but not the same as) support plate signals at the new balance point. In EXPECT (but not ANDES) there is compensation for U-bend drift because that part of EXPECT that extracts the individual events from the strip chart information measures each signal from its local balance point (refer to the comment statements in EVENT.LSP, starting on page F-21). Then, the U-bend drift (and the change in balance point labelled X + Y position ... on the strip charts) would not be visible on the EXPECT X-Y plots. The only unresolved parameter in U-bend drift is the amount by which the balance point changes and this can be determined by testing with a large number of tubes.

3.4.4 Wall Loss

This can be characterized as pits, cracks, erosion and wastage corrosion. They all have the same effect on tube integrity but have different causes. Wall loss can occur on the inside (Inner Diameter or ID) or outside (Outer Diameter or OD) the tube wall. It is very difficult to measure ID wall loss with any accuracy because of the steepness of the ID line in the wall loss vs. phase angle plot of Figure 3.2. Typically it is measured to the nearest 25% while OD wall loss is measured to the nearest 10%. Pits usually come in groups, sometimes many hundreds together and cracks are usually isolated. Wall loss can also occur due to the rubbing of the support plate against the tube (fretting - see 3.4.12). Wastage corrosion is wall loss spread over a large area. It has the same signature in the X-Y plots but has a much longer duration on the strip charts. It is important to distinguish among the different types of wall loss because they are usually caused by different phenomena. When the cause is known it sometimes can be eliminated or at least taken into account the next time a similar system is designed. For example, erosion can be caused by too much grit in the water, corrosion is caused by the wrong chemical make-up of the water and fretting can be caused by water flowing at some resonance frequency that has the effect of inducing vibrations in the structure. It is also important to know where wall loss occurs since some sections of the tube can be patched.

Isolated wall loss has been implemented in EXPECT by three rules, one each for inner diameter wall loss, holes and outer diameter wall loss. The rules are shown in Figure 3.12.

PROCEDURES USED IN RULE 3 - OUTER DIAMETER WALL LOSS

NON-NOSP: Refer to Rule 9, Section 3.4.1.

CURVE-TO--X?: True is returned if both CURVE-TO--X-OUT? and CURVE-TO--X-BACK? are true. These variables are true if the out and back signals (respectively) curve in a counterclockwise direction as the signal is traced from the balance point to the tip (page G-5).

ANGLE-TO-TIP-OD?: When TIP-ANGLE is between 44 and 110 degrees with respect to the -X axis, then this predicate returns true. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1 (page G-5).

O=B?: This is not to be confused with the predicate O=B of rule 1. In general, the defect signals are much smaller than the support plate signals, resulting in much less resolution in the expansion of the signals in terms of equal length basis vectors (see Section 3.5). In fact, using the same size basis vectors results in too much loss of information about the defects. Therefore, after all the support plate signals have been identified, EXPECT recalculates the expansion with the basis length reduced (by a factor of three for the results in this report). The predicate O=B? then returns T if the variable NN-O=B? is true. NN-O=B? is true if the out and back signals, expanded in terms of the reduced basis vectors, are the same (page G-5).

OD-ANALYSIS: If called, returns "OD - ##%", where ##, a measure of the wall loss, is a number between 10 and 90, in multiples of 10, or 5 if the indication is marginal (page G-6).

```
RULE 3      [OD-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE OUT OR BACK SIGNAL CURVES TOWARDS -X
      [CURVE-TO--X?]
  AND THE ANGLE TO TIP IS BETWEEN  44 AND 110 DEGREES WITH RESPECT TO THE -X
      AXIS
      [ANGLE-TO-TIP-OD?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]

 THEN
      THE EVENT IS DUE TO AN OUTER DIAMETER WALL LOSS OF ## %
      [OD-ANALYSIS]

RULE 4      [ID-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE OUT OR BACK SIGNAL CURVES TOWARDS -X
      [CURVE-TO--X?]
  AND THE ANGLE TO TIP IS BETWEEN  15 AND 35 DEGREES WITH RESPECT TO THE -X
      AXIS
      [ANGLE-TO-TIP-ID?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]

 THEN
      THE EVENT IS DUE TO AN INNER DIAMETER WALL LOSS OF ## %
      [ID-ANALYSIS]

RULE 5      [HOLE-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE OUT OR BACK SIGNAL CURVES TOWARDS -X
      [CURVE-TO--X?]
  AND THE ANGLE TO TIP IS BETWEEN  35 AND 44 DEGREES WITH RESPECT TO THE -X
      AXIS
      [ANGLE-TO-TIP-HOLE?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]

 THEN
      THE EVENT IS DUE TO A HOLE
      [HOLE-ANALYSIS]
```

Figure 3.12 **Rules 3,4 and 5 are used for the identification of wall loss. The angles, 15, 35, 44, and 110 degrees were taken from Figure 3.2 for the theta = 70° case (the bottom curve in the OD Quadrant). In general the angles will depend on the calibration. The symbol ## in the OD and ID rules indicates the amount of wall loss in percent and is calculated by the THEN procedure. Refer to the caption of Figure 3.10 for the interpretation of these rules.**

PROCEDURES USED IN RULE 4 - INNER DIAMETER WALL LOSS

NON-NOSP: Refer to Rule 9, Section 3.4.1.

CURVE-TO--X?: See Rule 3 - this Section.

ANGLE-TO-TIP-ID?: When TIP-ANGLE is between 15 and 35 degrees with respect to the -X axis, then this predicate returns true. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1 (page G-7).

O=B?: See Rule 3 - this Section.

ID-ANALYSIS: If called returns "ID - ##%", where ##, a measure of the wall loss, is a number between 25 and 75, in multiples of 25, or 10 if the indication is marginal (page G-7).

PROCEDURES USED IN RULE 5 - HOLE

NON-NOSP: Refer to Rule 9, Section 3.4.1.

CURVE-TO--X?: See Rule 3 - this Section.

ANGLE-TO-TIP-HOLE?: When TIP-ANGLE is between 35 and 44 degrees with respect to the -X axis, then this predicate returns true. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1 (page G-8).

O=B?: See Rule 3 - this Section.

HOLE-ANALYSIS: If called returns "HOLE" (page G-8).

3.4.5 Whalies

Whalies are produced during the manufacturing of the tube when the tube formation process is interupted for some reason. The effect on the tube is to produce a very small ID and OD wall loss.

Whalies appear similar to isolated wall loss except that the X-Y signals have the characteristic straightness (at all frequencies) as opposed to the counterclockwise curvature of wall loss. Another very characteristic feature that analysts always look for (which is not yet implemented but will be in any mature EXPECT) is the width of the signal on the strip charts. It is usually much (ten times) wider than a support plate or wall loss signal. The rule is shown in Figure 3.13.

```
RULE 7      [WHAL-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE LENGTH TO TIP IS < OR = HALF THAT OF THE AVERAGE NOSP LENGTH TO
      TIP
      [HALF-LEN-TO-TIP?]
  AND THE SIGNAL IS STRAIGHT
      [STRAIGHT?]

  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]
  AND THE SIGNAL IS IN THE SECOND QUADRANT
      [IN-QUAD-2?]

 THEN
      THE EVENT IS DUE TO A WHALIE
      [WHALIE-ANALYSIS]
```

- - + - -

Figure 3.13 Rule used for the identification of Whalies. Refer to the caption of Figure 3.10 for the interpretation of this rule.

PROCEDURES USED IN RULE 7 - WHALIES

NON-NOSP: Refer to rule 9, Section 3.4.1.

HALF-LEN-TO-TIP?: True is returned if TIP-LENGTH is less than half NOSP-AVG-TIP-LENGTH. TIP-LENGTH is the length of a line from the balance point to the tip. NOSP-AVG-TIP-LENGTH is the average value of TIP-LENGTH over the events that passed Rule 2 (Section 3.4.1) (page G-11).

STRAIGHT?: See Rule 6 - Section 3.4.6.

O=B?: See Rule 3 - Section 3.4.4.

IN-QUAD-2?: Returns true if TIP-ANGLE is in the second quadrant. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1 (page G-11).

WHALIE-ANALYSIS: If called returns "WHALIE".

3.4.6 Mechanical Deformation

This is a dent in the tube. It can indicate a manufacturing problem or the start of a vibration problem. Its X-Y signature is a line along the X axis, curving slightly clockwise. It most frequently occurs at the transition between a straight section and the u-bend, where it is the result of the manufacturing process. If the mechanical deformation is big enough then the probe cannot pass through.

```
RULE 6      [DENT-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE ANGLE TO TIP IS WITHIN + OR - 10 DEGREES OF THE X AXIS
      [WITHIN-10?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]
  AND THE SIGNAL IS STRAIGHT
      [STRAIGHT?]

 THEN
      THE EVENT IS DUE TO A DENT
      [DENT-ANALYSIS]
```

- - + - -

Figure 3.14 Rule used for the identification of Mechanical Deformation. Refer to the caption of Figure 3.10 for the interpretation of this rule.

In this version of EXPECT mechanical deformation is called a dent, which is in fact a different effect (17. in Table 3.1). Figure 3.14 shows the rule for identifying mechanical deformation.

PROCEDURES USED IN RULE 6 - MECHANICAL DEFORMATION

NON-NOSP: Refer to Rule 9, Section 3.4.1.

WITHIN-10?: Returns true if TIP-ANGLE is within 10° of the +X axis. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1 (page G-9).

O=B?: Refer to Rule 3 - Section 3.4.4.

STRAIGHT?: This predicate returns true if the signal is straight. This is determined by looking at the average angle of the basis vectors in the half of the signal from the balance point to the tip. If it is within 2° of the of the TIP-ANGLE then it is deemed straight. In retrospect it can been seen that this won't work very well. The algorithm should have been: the average of the absolute value of the deviation from the TIP-ANGLE is <= 2° (page G-9).

DENT-ANALYSIS: If called returns "DENT" (page G-10).

3.4.7 Deposit

This can occur near the tube sheet, near the support plate or along the length of the tube. There was not enough information to develop an analysis.

3.4.8 Resistivity Variations

This sometimes produces a signal in the defect quadrant but, typically, it is of longer duration on the strip chart and sometimes it is cyclic along the tube. It effect is to move the balance point. There was not enough information to develop an analysis.

3.4.9 Magnetic Permeability Changes

There was not enough information to develop an analysis.

3.4.10 Intergranular Attack

Intergranular attack is corrosion along grain boundaries in the tube wall. This can be very serious but in practice it cannot be detected because very little material is removed.

3.4.11 Support Plate Plus Wall Loss

The wall loss can be due to pits, cracks etc. and occur anywhere near or under the support plate. Figures 3.4 - 3.6 show what this type of defect looks like.

A rule to analyze this situation for the special case where the wall loss is at the support plate centre (as for events 6 in Figures 3.4 - 3.6) has been implemented. This Rule is of the recognize-and-analyze type but, as discussed in Section 3.3, the general case where the wall loss can be anywhere under the support plate is better analyzed by the guess-and-test method. Figure 3.15 shows the Rule.

PROCEDURES USED IN RULE 8 - WALL LOSS AT THE SUPPORT PLATE CENTRE

PROSP-TEST: Refer to Rule 2 - Section 3.4.1.

NON-NOSP: Refer to Rule 9, Section 3.4.1.

LONGER-LEN-TO-TIP?: Returns true if the TIP-LENGTH is more than 10% longer than NOSP-AVG-TIP-LENGTH. TIP-LENGTH is the length of a line from the balance point to the tip. NOSP-AVG-TIP-LENGTH is the average value of TIP-LENGTH over the events that passed Rule 2 (page G-12).

```
RULE 8      [WLSPC-ID.LSP]

 IF
      THE SIGNAL IS A PROSP
      [PROSP-TEST]
  AND THE SIGNAL IS NOT A NOSP
      [NON-NOSP]
  AND THE TIP LENGTH IS MORE THAN 10% LONGER THAN THE AVERAGE NOSP TIP
      LENGTH
      [LONGER-LEN-TO-TIP?]

 THEN
      THERE IS WALL LOSS (## %) AT THE SUPPORT PLATE CENTER
      [WALL-LOSS-AT-SP-CENTER]
```

- - + - -

Figure 3.15 Rule used for the identification of support plate plus wall loss at the support plate centre. Refer to the caption of Figure 3.10 for the interpretation of this rule.

WALL-LOSS-AT-SP-CENTER: If called, returns "sp-OD-##%", "sp-ID-##%" or "sp-HOLE", depending on whether the wall loss is caused by an OD, ID or hole fault. ## is the amount of wall loss in percent. The angle represented by the extra length of the signal is the angle between the tip and a new balance point. The new balance point is the point of the signal where the distance to the old balance point equals NOSP-AVG-TIP-LENGTH (see previous predicate). The wall loss is then found from the angle using Figure 3.2, as in Section 3.4.4. Without considerable testing it is difficult to know if this algorithm will work in all cases (page G-12).

3.4.12 Fretting at a Support Plate

Fretting is the wearing away of material under the support plate due to vibrations. The frets can be either symmetric or asymmetric (see Figure 3.16) with the symmetric frets rotating the support plate signal clockwise by an amount that depends on the amount of material removed and the asymmetric frets looking like a pit at the support plate edge. To find the difference between an asymmetric fret and a pit at the edge of a support plate, an analyst would look at the whole tube or the history of the heat exchanger. For example, if corrosion has been a problem then there will be corrosion all along the tube but if every OD indication is at a support plate then it is probably fretting. Fretting is usually not as serious as corrosion or erosion but the analyst will always want to identify it.

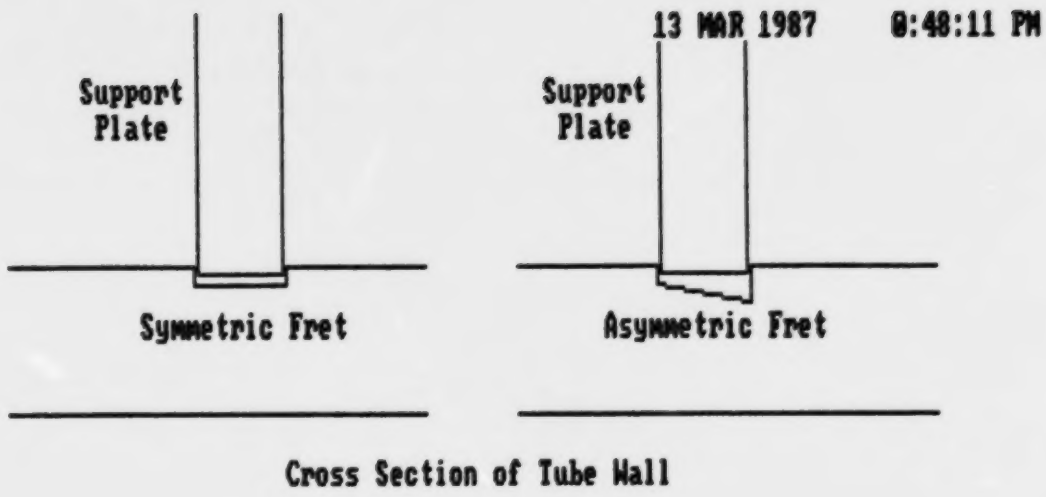


Figure 3.16 Symmetric and asymmetric fret at a support plate site.

A rule to analyze this situation for the special case where the fret is symmetric is shown in Figure 3.17. This rule is of the recognize-and-analyze type but, the general case where the fret can be asymmetric and looks like a pit at the support plate edge is better analyzed by the guess-and-test method (Section 3.3).

PROCEDURES USED IN RULE 12 - SYMMETRIC FRET AT A SUPPORT PLATE

ANY-NOSPS?: This predicate returns true if there have been any normal support plates identified. The principal characteristic of the symmetric fret is that the signal looks like a support plate signal except that it is rotated clockwise with respect to the normal support plate signals. If there are no normal support plate signals then there is no purpose in evaluating the other predicates (page G-19).

```
RULE 12     [SFRET-ID.LSP]

 IF

      THIS TUBE HAS SOME NORMAL SUPPORT PLATE SIGNALS
      [ANY-NOSPS?]
  AND THE SIGNAL IS NOT A NORMAL SUPPORT PLATE SIGNAL
      [NON-NOSP]
  AND THE INITIAL ANGLE (BOTH OUT AND BACK) IS IN THE 1ST OR 2ND QUADRANT
      [IA-Q]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B]
  AND THE S SHAPE IS PRESENT IN BOTH THE OUT AND BACK SIGNALS
      [S-SHAPE]
  AND THE SIGNAL IS ROTATED CLOCKWISE FROM THE NOSP AVERAGE TIP ANGLE
      [ROTATED-UP?]
  AND THE LENGTH IS NOT LESS THAN THE NOSP AVERAGE TIP LENGTH
      [LENGTH>=?]

 THEN
      THE EVENT IS DUE TO A SYMMETRIC FRET AT A SUPPORT PLATE WITH
      WALL LOSS OF ## %
      [SYMMETRIC-FRET-ANALYSIS]
```

\- - + - -

Figure 3.17 Rule used for the identification of a symmetric fret at a support plate. Refer to the caption of Figure 3.10 for the interpretation of this rule.

NON-NOSP: Refer to Rule 9, Section 3.4.1.

IA-Q: Refer to Rule 1 - Section 3.4.1.

O=B: Refer to Rule 1 - Section 3.4.1.

S-SHAPE: Refer to Rule 1 - Section 3.4.1.

ROTATED-UP?: True is returned when TIP-ANGLE is rotated more than NOSP-ANGLE counterclockwise with respect to NOSP-AVG-TIP-ANGLE. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1. NOSP-ANGLE is a constant (5° for the results in this report). NOSP-AVG-TIP-ANGLE is the average value of TIP-ANGLE for the events that passed Rule 2 (page G-19).

LENGTH>=?: Returns true if TIP-LENGTH is longer than NOSP-AVG-TIP-LENGTH. TIP-LENGTH is the length of a line from the balance point to the tip. NOSP-AVG-TIP-LENGTH is the average value of TIP-LENGTH over the events that passed Rule 2 (page G-19).

SYMMETRIC FRET ANALYSIS: If called returns "SF-##%", where ## represents the degree of wall loss. The wall loss is found from the amount of counterclockwise rotation and Figure 3.2, as in Section 3.4.4 (page G-20).

3.4.13 Tube Sheet Plus Wall Loss

The wall loss component pulls the tube sheet signal into the defect quadrant. This is usually difficult to analyze because there is no tube sheet return signal and the signal saturates.

Section 3.4.2 contains a discussion of the characteristics of tube sheet plus wall loss and tube sheet plus mechanical deformation. Since the part of the tube sheet signal that is important (initial angle) is relatively easy to measure, it should be a simple matter to detect defects. The only difficulty may be that, relative to the support plate, there far fewer signals for comparison.

3.4.14 Support Plate Plus Mechanical Deformation

The dent tends to pull the support plate signal into the 1st quadrant. This is a good candidate for the guess-and-test techniques of Section 3.3.

3.4.15 Pit Within a Pit

Figure 3.18 shows a sketch of this type of defect. The X-Y signal starts at one angle corresponding to the shallow pit and at the tip of the signal (when the 2nd pit is at the centre of the first as shown in the Figure) there is a discontinuity in slope corresponding to the depth of the second pit. Figure 3.19 shows an example of the X-Y response. Event 7 corresponds to the symmetric case of Figure 3.18 and the discontinuity of slope can be seen. It is obvious that it is important to recognize both signals since it is the total depth of the wall loss that determines its seriousness. It would seem very difficult to find the hole in events 5,6,8 and 9 of Figure 3.19. In fact any decomposition of these events is probably non-unique. The on-line software tool described in Section 3.3 would be of great assistance in cases such as this one.

3.4.16. Dents

This is a different problem from mechanical deformation of Section 3.4.6. It is caused by the wearing away of the edge of a support plate and the formation of oxides. The oxides swell and deform the tube. This situation would look like a support plate signal plus mechanical deformation: the support plate signal would be pulled along the X axis towards the 1st quadrant. To complicate this further there could be an associated defect as well. The situation of support plate, dent and defect would benefit greatly from the software tool described in Section 3.3.

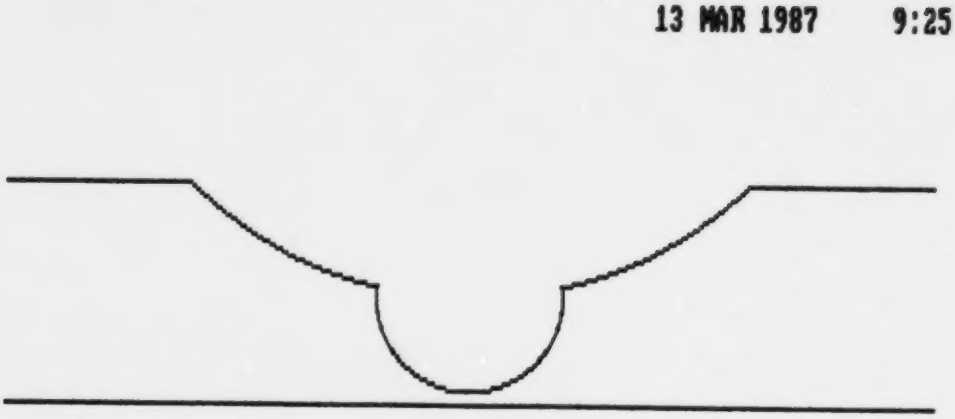


Figure 3.18 Cross section of a tube wall showing sketch of the pit within a pit type of defect. This corresponds to the symmetric case (by far the most common) where the second pit is in the centre of the first pit.

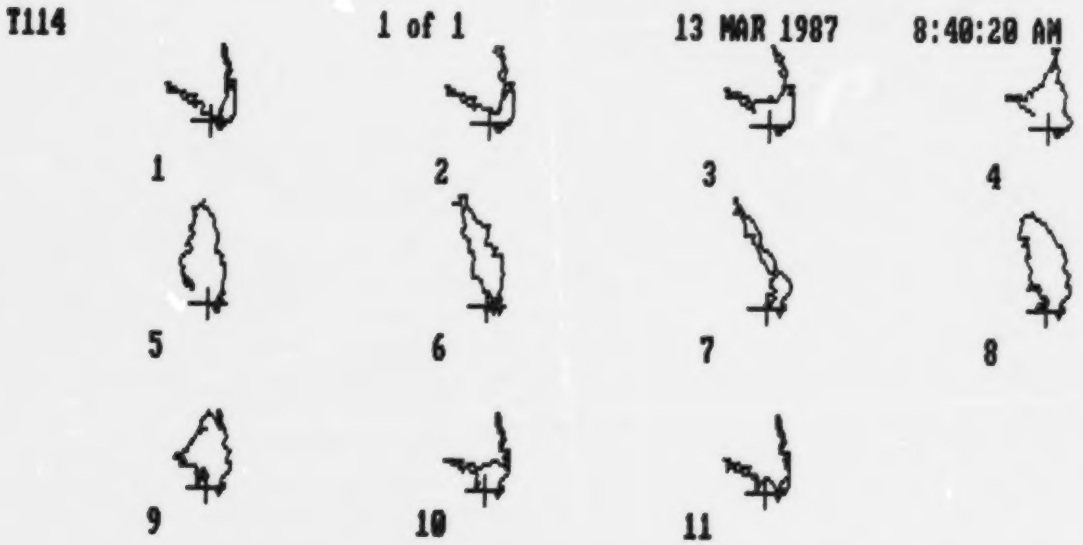


Figure 3.19 The addition of two defect signals: a 30% OD signal and a hole signal. The 30% OD signal is that of event 2 in Figure 3.1 and the hole signal is the one used in Figures 3.5 and 3.6. The different events represent different relative displacements of the two defects. Event 7 roughly corresponds to the situation sketched in Figure 3.18, and events 1 and 11 to separated defects.

3.4.17 Other Combinations

In practice any combination of defects, at or away from a normal signal can occur. It seems quite difficult to automate the analysis process for arbitrary combinations. However, see the discussion at the end of Section 3.3.

3.4.18 Change in Balance Point

Sometimes the operator will move the balance point to obtain a more effective dynamic range for the signal (see Figure 3.8). The procedures in EXPECT keep track of this although it is not explicitly displayed.

3.4.19 Probe Wobble

The probe is centered in the tube by means of guides that touch the walls. Sometimes the guides get worn down and the probe bounces up and down as it moves along the tube. This results in a back and forth oscillation in the X-Y signal along the X axis. It may be possible to detect this using Fourier analysis techniques. This requires further work.

3.4.20 Characteristics of ANDES

Sometimes when two signals are close together ANDES will not separate them. This could look like Event 1 in Figures 3.4 - 3.6 or Figure 3.19. Andes occasionally separates the two halves of a support plate signal. This topic has not been investigated thoroughly and there may or may not be other characteristics of ANDES that EXPECT has to be able to identify. Or it may be that the latest version of ANDES has no characteristics that need to be taken into account. Figure 3.20 shows a rule for identifying half of a support plate signal.

PROCEDURES USED IN RULE 11 - HALF A SUPPORT PLATE SIGNAL

NON-NOSP: Refer to Rule 9, Section 3.4.1.

IA-Q-OR?: If either the INITIAL-ANGLE-OUT or INITIAL-ANGLE-BACK is in the first or second quadrant, then true is returned. Refer to IA-Q, Rule 1, Section 3.4.1 (page G-17).

S-SHAPE-OR?: True is returned if either the out or the return signals have an "S" shape. Refer to S-SHAPE, Rule 1, Section 3.4.1 (page G-17).

MIN-X-ANGLE?: The tip of the signal has to be defined a little differently since there is only half a signal: the point of minimum X of the signal. This predicate tests to see if the TIP-ANGLE is within NOSP-ANGLE of NOSP-AVG-TIP-ANGLE. TIP-ANGLE is the angle in degrees with respect to the -X axis of a line from the balance point to the tip - see Figure 3.1. NOSP-ANGLE is a constant (5° for the results in this report). NOSP-AVG-TIP-ANGLE is the average value of TIP-ANGLE for the events that passed Rule 2 (page G-18).

```
RULE 11      [HAFSP-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE OUT OR BACK SIGNAL INITIAL ANGLE IS IN THE 1ST OR 2ND QUADRANT
      [IA-Q-OR?]
  AND THE OUT OR BACK SIGNAL HAS AN S-SHAPE
      [S-SHAPE-OR?]
  AND THE SIGNAL TIP ANGLE (MEASURED FROM THE MINIMUM X POINT) IS WITHIN
      nosp-angle DEGREES OF THE NOSP AVERAGE TIP ANGLE
      [MIN-X-ANGLE?]
  AND THE SIGNAL TIP LENGTH (MEASURED FROM THE MINIMUM X POINT) IS WITHIN
      nosp-length (FRACTION) OF THE NOSP AVERAGE TIP LENGTH
      [MIN-X-LENGTH?]

 THEN
      THE EVENT IS DUE TO A HALF SUPPORT SIGNAL
      [HALF-SUPPORT-PLATE-ANALYSIS]
```

\- - + - -

FIGURE 3.20 Rule used for the identification of half of a support plate signal. Refer to the caption of Figure 3.10 for the interpretation of this rule.

MIN-X-LENGTH?: Returns true if TIP-LENGTH (with the tip defined as in the previous predicate) is within NOSP-LENGTH of NOSP-AVG-TIP-LENGTH. TIP-LENGTH is the length of a line from the balance point to the tip. NOSP-AVG-TIP-LENGTH is the average TIP-LENGTH for all those events that were identified as normal support plates. NOSP-LENGTH is a fraction (of NOSP-AVG-TIP-LENGTH), with .1 being the value used for the results in this report (page G-18).

HALF-SUPPORT-PLATE-ANALYSIS: If called returns "HALF SP" (page G-18).

3.4.21 Saturated Signal

This refers to the signal being larger than the dynamic range of the digitizer and shows up on the X-Y trace as a straight line at one of the 4 edges. Figure 3.1, event 11 shows an example of this. Figure 3.21 shows a rule for identifying a saturated signal.

```
RULE 10     [SAT-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THERE ARE MORE THAN 20 VALUES OF ONE OF THE FOLLOWING: MAXIMUM X,
      MINIMUM X, MAXIMUM Y OR MINIMUM Y
      [TOO-MANY-EDGE-VALUES?]

 THEN
      THE SIGNAL IS SATURATED
      [SATURATION-ANALYSIS]
```

- - + - -

FIGURE 3.21 Rule used for the identification of a saturated signal. Refer to the caption of Figure 3.10 for the interpretation of this rule.

PROCEDURES USED IN RULE 10 - SATURATED SIGNAL

NON-NOSP: Refer to Rule 8, Section 3.4.1.

TOO-MANY-EDGE-VALUES?: This predicate looks for an accumulation of values on the four edges (of the X-Y plot) and if there are more than 20 then the signal is considered saturated. An accumulation in defined as the number of maximum values plus (maximum - 1) values plus (maximum - 2) values; and similarly for the minimum edges (page G-16).

SATURATION-ANALYSIS: If called returns " SAT" (page G-16).

3.4.22 Die Chatter

These are marks in the tube that originate during the manufacturing process. Their wavelength can vary along the tube and from tube to tube. It looks much like probe wobble and if the signals are large enough, can mask a defect signal. It may be possible to detect this with Fourier analysis.

3.5 EXPECT Variables

In the predicate descriptions of the previous Section, EXPECT LISP variables were given in upper case letters (an example is TIP-LENGTH). This Section explains how these variable were calculated by EXPECT.

However, before this is done it is necessary to understand how the X-Y signals are represented internally in EXPECT. ANDES produces a list of X - Y pairs for each event, such as those shown plotted in Figure 3.1. For Event 1 in the Figure there are 140 pairs and for Event 2, 77 pairs. The number of features that we wish to extract (ie those that the analysts uses), is much smaller than the number of X-Y pairs. For example, the isolated wall loss (Event 2) is recognized by its counterclockwise curvature, the angle to the tip being in a certain range, and the out and return signals being the same. Therefore, the 77 pairs of numbers have to be reduced to three binary values.

The first step is to find the tip of the signal:

1. Find the maximum distance along the signal, from the balance point.

2. Find the point along the out half of the signal where the distance is 90% of the maximum distance. Do the same for the back half of the signal.

3. If these two points represent the i^{th} and j^{th} elements in the lists of X & Y values then the tip is the $(i + j)/2^{th}$ element.

This is a little complicated but works where simpler algorithms will not. For example, if the tip were to be defined as the element in the list of pairs that has the minimum X value then it would give about the same answer as the above for Event 1 of Figure 3.1 but unlike the above would not find the tip for event 2. Once the tip is found then the angle from the balance point (TIP-ANGLE) to the tip can be calculated.

The next step is to smooth out the signals so that the gross features, such as the "S" shape, can be found. This is done by fitting the signals with basis vectors having fixed length, and quantized angles. For the examples given in this report the fixed length was 10% of the absolute value of the minimum X value over all the events and the angles were integer multiples of 360°/64.

Figures 3.22 and 3.23 show an expansion in terms of these basis vectors for Events 1 and 2 (respectively) of Figure 3.1. Note that because the first vector in the expansion for both the out and back halves of the signal start at the balance point, the end of the last vector in each half does not, in general, end at the tip. The leftmost diagram in each Figure reproduces (with more resolution), Events 1 and 2 of Figure 3.1. The next diagram shows the same signal expanded in terms of the basis vectors. It can be seen that for Event 2, with only 2 basis functions, there is not enough resolution to properly match the shape of the signal. For this reason all non-normal support plates signals are re-expanded in terms of another set of basis functions, with the same set of angles but with a length 1/3 that of the original basis functions. The result of this expansion is shown in the rightmost diagram of Figure 3.23.

Figure 3.22 On the left is Event 1 of Figure 3.1, shown with more resolution. At the right is the same signal, expanded in terms of the basis functions, as explained in the text.

Figure 3.23 On the left is Event 2 of Figure 3.2, shown with more resolution. In the centre is the same signal, expanded in terms of the same set of basis functions as in Figure 3.22. On the right is the same signal expanded in terms of a set of basis functions with a smaller length - see the text.

```
FILE NAME  T-1-11
EVENT  1
TIP-INDEX  65
TIP-ANGLE  -3
TIP-LENGTH  198.305
VECTOR-ENDS-OUT  (13 20 24 27 30 33 36 41)
VECTOR-ENDS-BACK  (113 108 104 101 98 95 92 87)
BASIS-OUT     (19 32 35 35 36 35 33 31)
              (- + + + + + + -)
BASIS-BACK    (25 30 32 34 37 35 35 35)
              (- - + + + + + +)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  107
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (1 4)
S-SHAPE-BACK? (2 4)
OUT=BACK?  T
PROSP?  T
NOSP?  T
ANALYSIS  sp
```

Figure 3.24 Some calculated properties for Event 1 of Figure 3.1. Refer to the text for an explanation.

```
FILE NAME  T-1-11
EVENT  2
TIP-INDEX  28
TIP-ANGLE  98
TIP-LENGTH  44.4072
VECTOR-ENDS-OUT  (16 24)
VECTOR-ENDS-BACK  (41 32)
BASIS-OUT     (10 19)
              (- -)
BASIS-BACK    (10 20)
              (- -)
BASIS-LENGTH-OUT  2
BASIS-LENGTH-BACK 2
INITIAL-ANGLE-OUT  56
INITIAL-ANGLE-BACK 56
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  T
PROSP?  NIL
NOSP?  NIL
NN-VE-OUT  (0 12 14 18 21 26)
NN-VE-BACK  (62 49 45 41 38 37 33)
NN-B-OUT  (61 13 11 20 17 19)
NN-B-BACK  (58 8 12 16 24 15 18)
NN-BL-OUT  6
NN-BL-BACK  7
NN-IA-OUT  343
NN-IA-BACK  326
NN-O=B?  T
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  30
ANALYSIS  (OD - 30%  SAT)
```

Figure 3.25 Some calculated properties for Event 2 of Figure 3.1. Refer to the text for an explanation.

Figures 3.24 and 3.25 show the numerical value of several EXPECT variables in the expansion for Events 1 and 2 of Figure 3.1. The results for the rest of the events for this tube are in Appendix A and the results for the rest of the Incoloy 800 tubes are in Appendix J. Appendix B shows a listing of the EXPECT internal data structure where all the information on tube T-1-11 is stored. The variables BASIS-OUT, BASIS-BACK, NN-B-OUT and NN-B-BACK are lists of angles in units of 360°/64 i.e. 0 is 0°, 16 is 90°, 32 is 180° etc. BASIS-OUT AND BASIS-BACK represent the expansion in terms of the longer length (Figure 3.22 and middle diagram of Figure 3.23) and NN-B-OUT and NN-B-BACK in terms of the shorter length (rightmost diagram of Figure 3.23). Note that because Event 1 is a normal support plate signal, there was no expansion in terms of the shorter length. The lists of pluses and minuses below the lists of angles indicates whether the corresponding element of the list of angles is above 180° (+) or below it. Since the length of the expansion vectors is constant then the list of angles is a complete representation of the signals. The rest of the EXPECT variables are explained below. The concepts are explained below in english and reference is given to the LISP code where the algorithm is explicit.

ANALYSIS: EXPECT's diagnosis of the signal.

Balance point: In EXPECT the balance point is the inter-event zero point (see Figure 3.1). It marks the local base of the signal peak and could change along the tube to match an adjustment to the X or Y strip chart signals. These adjustments would then not show on the X-Y plots (although the information is stored internally in EXPECT). In ANDES the balance point stays fixed throughout the tube so that X or Y position adjustments are visible on the X-Y plots (see the upper half of Figure 3.8).

BASIS-BACK: See VECTOR-ENDS-BACK.

BASIS-LENGTH-BACK: The number of elements of VECTOR-ENDS-BACK.

BASIS-LENGTH-OUT: The number of elements of VECTOR-ENDS-OUT.

BASIS-OUT: See VECTOR-ENDS-OUT.

CURVE-TO--X-BACK?: This is true if the average of the angles in the first half of NN-B-BACK? is less than that of the second half. Note that the branch point had to be changed from 0° to -90° so that angles between 270° and 360° would become -90° to 0°. This fiddle allows the program to identify counterclockwise curvature quickly for the range of values expected.

CURVE-TO--X-OUT?: The same as CURVE-TO--X-BACK? except that it refers to the out half of the signal.

INITIAL-ANGLE-BACK: The first element of BASIS-BACK, in degrees with respect to the X axis.

INITIAL-ANGLE-OUT: The first element of BASIS-OUT, in degrees with respect to the X axis.

NOSP?: True if the Event passes Rule 2. NOSP stands for normal support plate.

NOSP-ANGLE: See the predicate PROSP-AVG-ANGLE in Rule 2.

NOSP-AVG-TIP-ANGLE: Average TIP-ANGLE of those Events identified as NOSPs by Rule 2.

NOSP-AVG-TIP-LENGTH: Average TIP-LENGTH of those Events identified as NOSPs by Rule 2.

NOSP-LENGTH: A constant equal to a fraction of PROSP-AVG-TIP-LENGTH - see RULE 2 for usage. The value used for the results in this report is .1.

NN-B-BACK: The same as BASIS-BACK except the second expansion (in terms of the shorter basis vectors) is used.

NN-B-OUT: The same as BASIS-OUT except the second expansion (in terms of the shorter basis vectors) is used.

NN-BL-BACK: The same as BASIS-LENGTH-BACK except the second expansion (in terms of the shorter basis vectors) is used.

NN-BL-OUT: The same as BASIS-LENGTH-OUT except the second expansion (in terms of the shorter basis vectors) is used.

NN-IA-BACK:The same as INITIAL-ANGLE-BACK except the second expansion (in terms of the shorter basis vectors) is used.

NN-IA-OUT: The same as INITIAL-ANGLE-OUT except the second expansion (in terms of the shorter basis vectors) is used.

NN-O=B?: The same as OUT=BACK? except the second expansion (in terms of the shorter basis vectors) is used.

NN-VE-BACK: The same as VECTOR-ENDS-BACK except the second expansion (in terms of the shorter basis vectors) is used.

NN-VE-OUT: The same as VECTOR-ENDS-OUT except the second expansion (in terms of the shorter basis vectors) is used.

OUT=BACK?: This has a value of T (true), indicating that the out and back signals are approximately the same, or F, indicating that they are not. They are not the same if the number of elements in VECTOR-ENDS-OUT and VECTOR-ENDS-BACK differ by more than a constant (20%, for the results given in this report). They are the same if the sum of the squares of the distance between the end points of the corresponding basis vectors (ie each indexed from the balance point) is less than a constant (the square of the minimum X value in the tube, for the results given in this report). The number of terms in the sum is the minimum of the length of BASIS-OUT and BASIS-BACK.

PROSP?: True if the Event passes Rule 1. PROSP stands for prospective normal support plate.

PROSP-AVG-TIP-ANGLE: Average TIP-ANGLE of those Events identified as PROSPs by Rule 1.

PROSP-AVG-TIP-LENGTH: Average TIP-LENGTH of those Events identified as PROSPs by Rule 1.

S-SHAPE-BACK: This refers to the return signal. It can have a value of F, indicating that the return signal does not have an "S" shape (see Figure 3.25) or it can be a list of two numbers indicating that it does have an "S" shape. The signal has an "S" shape if it starts with an initial angle in the first or second quadrant, followed by a maximum and then an inflection point. The first number in the list is the element number of the first element in BASIS-OUT (the element numbers start at 0) that is equal to or larger than 32 (ie 180°). This indicates that the signal has gone through a maximum. The inflection point is present if, subsequently, the list of angles goes through a maximum. The element number of the maximum is the second number in the list. Refer to Figures 3.24 and 3.25 for examples that do and do not (respectively) have an "S" shape.

S-SHAPE-OUT: The same as S-SHAPE-BACK except that it refers to the first half of the signal.

TIP-ANGLE: The text at the beginning of this Section explains how the tip of a signal is found. TIP-ANGLE is the angle from the balance point to the tip.

TIP-INDEX: The text at the beginning of this Section explains how the tip of a signal is found. The TIP-INDEX is the element number in the lists of X and Y values of the tip.

TIP-LENGTH: The text at the beginning of this Section explains how the tip of a signal is found. TIP-LENGTH is the length from the balance point to the tip.

VECTOR-ENDS-BACK: The same as VECTOR-ENDS-OUT, except that it refers to the return from the tip to the balance point. The sequence is decreasing because the process is started from the balance point, which for the return signal is at the end of the list of X-Y pairs.

VECTOR-ENDS-OUT: To find the first basis vector, each value of X*X + Y*Y (with the balance point as the zero point) from the lists of X and Y values that make up the Event are calculated (starting with the first, and in sequence) until one pair is found where X*X + Y*Y is greater than the square of the length of the basis vectors. The element number of this pair is the first element of VECTOR-ENDS-OUT. The angle in the set of basis angles that is closest to the angle from the zero point to the pair is recorded in BASIS-OUT. This process is continued, with the newest member of VECTOR-ENDS-OUT being the new zero point, until the tip is reached (which in general is not included - see Figure 3.22). If a better fit is needed then it is simply a matter of shortening the length of the basis vectors.

WALL-LOSS: The wall loss in %.

4. RESULTS OF USING EXPECT WITH TUBE DATA

Table 4.1 shows the tubes on which EXPECT was tested. Figures 4.1 - 4.4 show the results of applying EXPECT to the Incoloy 800 tubes - the tubes that this report addresses. The captions of the Figures provide comments on the results. Appendix D shows the analysis provided by CNS for those tubes whose EXPECT label in the Table are prefixed by T. Figures 4.5 to 4.8 show the separate X and Y components for the tubes of Figures 4.1 to 4.4.

Appendix C shows the results of applying EXPECT to the rest of the tubes of Table 4.1. The first batch of tubes in Appendix C (T-CAL-1 to T-33-26) have a lot of problems, whereas the 01-01R to 03-20 group are problem free. EXPECT does reasonably well on these tubes. It seems to be able to pick out saturated signals that are not evident to the eye and would most likely be labelled by the analysts as something else (this may also mean that the rule is too sensitive and needs some adjustments). For example, events 5 and 28 of T-13-39 are probably support plates and event 11 of 01-01R, although the signal is a little saturated, is evidently a support plate signal. There is also evidence that the rule for a saturated signal is not sensitive enough: in tube T-CAL-3 some of the signals are obviously saturated. EXPECT also does well on the ID-OD-HOLE wall loss type of signal although it doesn't analyze them all correctly (event 14 of T-25-25 is probably an ID indication). Event 12 of T-25-25 looks somewhat like events 4 and 9 of Figure 3.6 so would be wall loss plus a support plate, although as discussed in Chapter 3 the present version of EXPECT cannot analyze signals such as this one.

Comparing the support plate signal of T-1-11, T-25-25 and 01-01R shows the diversity of support plate signals to which the rules of Chapter 4 were applied successfully. It is an indication of the robustness of the rules and the underlying algorithms that they do well on tubes made of materials for which they were not really intended.

Timing results are shown in Table 4.2. The programs were run on an 8 Mhz Compaq Portable 286 with a 5 Mhz 80287 numeric co-processor. Of concern to a final system is the speed with which the signals can be analyzed, not the speed with which the peaks can be separated since this is done by ANDES. The final system will have more rules (possibly many more) but this balanced against the fact the control structure will be better, the program will probably be run on a microcomputer based on the 80386 chip (the 20 Mhz 80386 is about 6 - 8 times faster than the 8 Mhz 80286 using the same LISP and same version of DOS) with a 80387 numeric co-processor (the 80387 will be about 8 to 12 times faster than the 5 Mhz 80287), and the number crunching sections of EXPECT will be translated into C which is much more efficient than LISP at this. In addition, there are specially designed LISP plug-in boards that will speed up the LISP code even more. Therefore, speed is not considered to be a problem at this point.

As noted in the caption of Figure 4.6, some of the peaks is the X - Y data were missed by EXPECT. Very little effort was put into the peak extraction aspect of EXPECT since it duplicates the ANDES program and it was felt that time could be spent better by exploring other matters. In the next phase it is anticipated that the EXPECT peak extraction code will be discarded and ANDES will be used directly.

Table 4.1 Tubes on which EXPECT was tested. Those tubes whose EXPECT label contain CAL are calibration tubes. In the remaining tubes, the numbers in the EXPECT label indicate the row and column of the tube.

EXPECT Label	Source	Type	Material	Size	Frequency
T-CAL-0	PNGS Unit 6	MOD HX-2	Incoloy 800	1/2" OD 0.049" Wall	220kHZ
T-1-11	"	"	"	"	"
T-1-22R	"	"	"	"	"
T-2-6	"	"	"	"	"
T-CAL-1	PNGS Unit 5	RCW HX 502	90/10 Cu/Ni	5/8" OD .049" Wall	25 kHZ
T-CAL-2	"	"	"	"	"
T-13-39	"	"	"	"	"
T-25-25	"	"	"	"	"
T-25-30	"	"	"	"	"
T-CAL-3	"	"	"	"	"
T-29-46	"	"	"	"	"
T-33-5	"	"	"	"	"
T-33-26	"	"	"	"	"
01-01R	PNGS Unit 5	Mod Hx N7058D	?	?	145 kHZ
01-05	"	"	"	"	"
01-10	"	"	"	"	"
01-15	"	"	"	"	"
01-20	"	"	"	"	"
03-25	PNGS Unit 6	Mod Hx N7058D	?	?	220 kHZ
03-20	"	"	"	"	"

Table 4.2 Timing results for the tubes of Figures 4.1 to 4.4. The raw data referred to in the second column is the ANDES data. The time to separate the signal is the time to separate out the peaks (see Figures 4.5 - 4.8) from the background data. This was done in LISP and is similar in purpose to the ANDES program. The time to analyze the signal was the time spent in analyzing the separated peaks to produce the EXPECT diagnosis (ie to fill the LIMITS, CALCULATED-PROPERTIES, NOSP, NON-NOSP and ANALYSIS slots of the TUBE DEFSTRUCT structure - see Chapter 2 and the program listings in the Appendixes). The column on the right contains the size of the file needed to store all the analyzed information about the tube (ie the size of the TUBE DEFSTRUCT structure stored in the T-CAL-0.S etc. files - see Chapter 2 and the program listings).

Tube	File Size Of Raw Data (Bytes)	Time To Separate The Signals (Seconds)	Time To Analyze The Signals (Seconds)	Size Of File Containing LISP Data Structure (Bytes)
T-CAL-0	57344	275	46	18979
T-1-11	69632	475	37	14828
T-1-22R	73727	551	23	10247
T-2-6	65526	511	40	17242

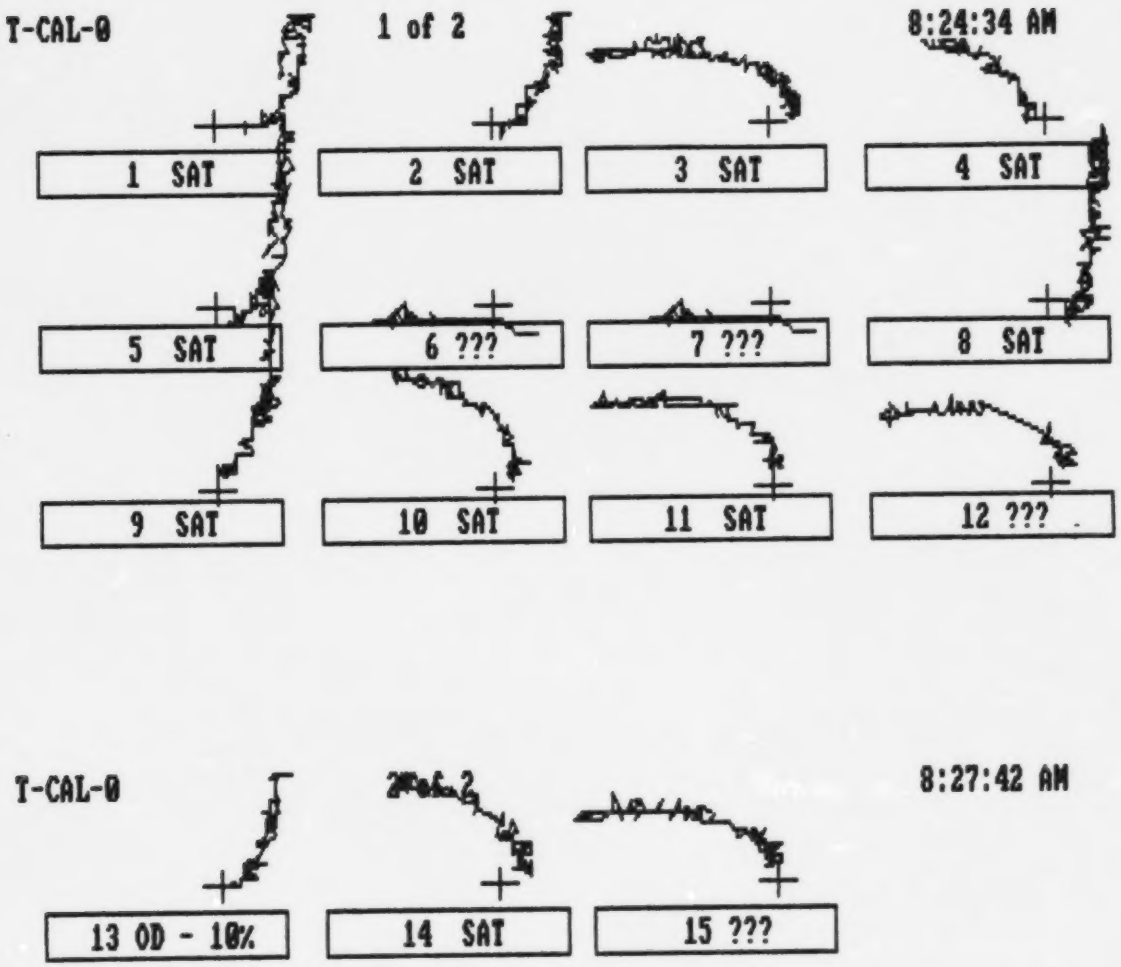


Figure 4.1 EXPECT results for T-CAL-0, a calibration tube. Most of the signals are saturated. It does not identify event 12 as wall loss because there is only half a signal (see the value of the NN-VE-OUT variable for event 12 for T-CAL-0 in Appendix J).

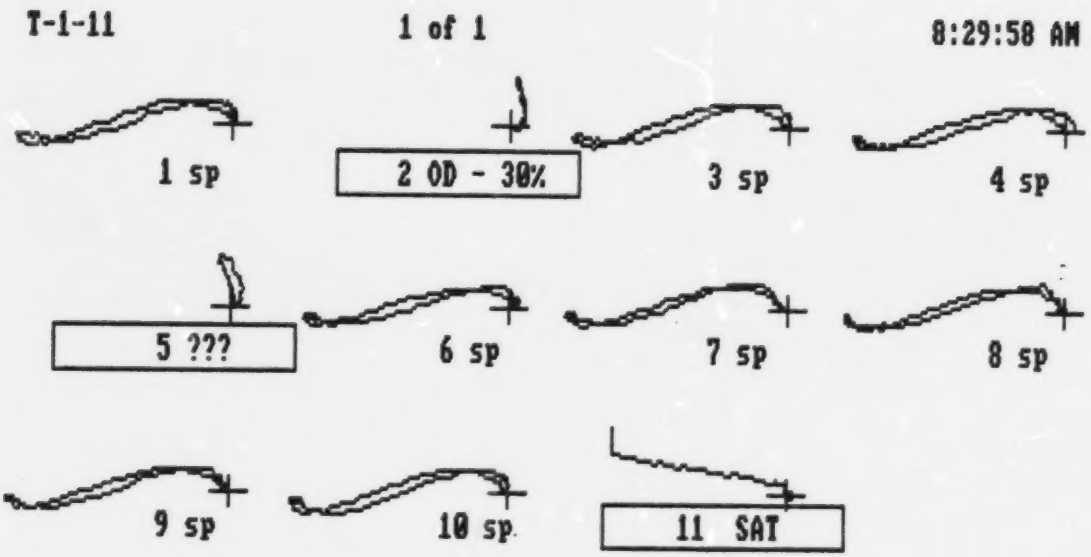


Figure 4.2 EXPECT results for T-1-11. Event 5 looks somewhat like events 6,7 and 8 in Figure 3.19; it was not identified as a wall loss signal because the signal going out to the tip was not the same as the signal returning to the balance point (refer to Section 3.5). Event 11 is part of the tube sheet signal. The CNS analysis of this signal is: 1. Small amplitude 25% to 40% OD, and 2. Possible ID wall loss. The former is event 2 and the latter may be event 5 but more likely: EXPECT did not separate the event from the signal. See the comments under Figure 4.6.

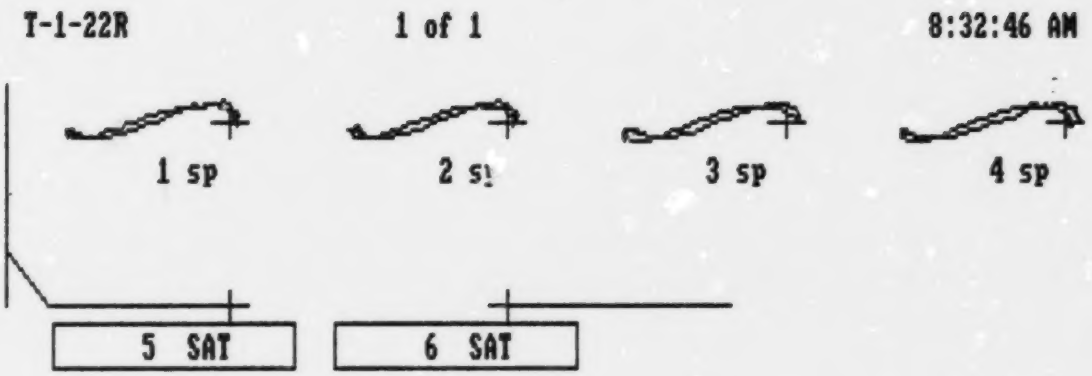


Figure 4.3 EXPECT results for T-1-22R. The CNS analysis of this tube was: large amplitude OD at tube sheet. There are no rules to identify composite signals at the tube sheet in this version of EXPECT.

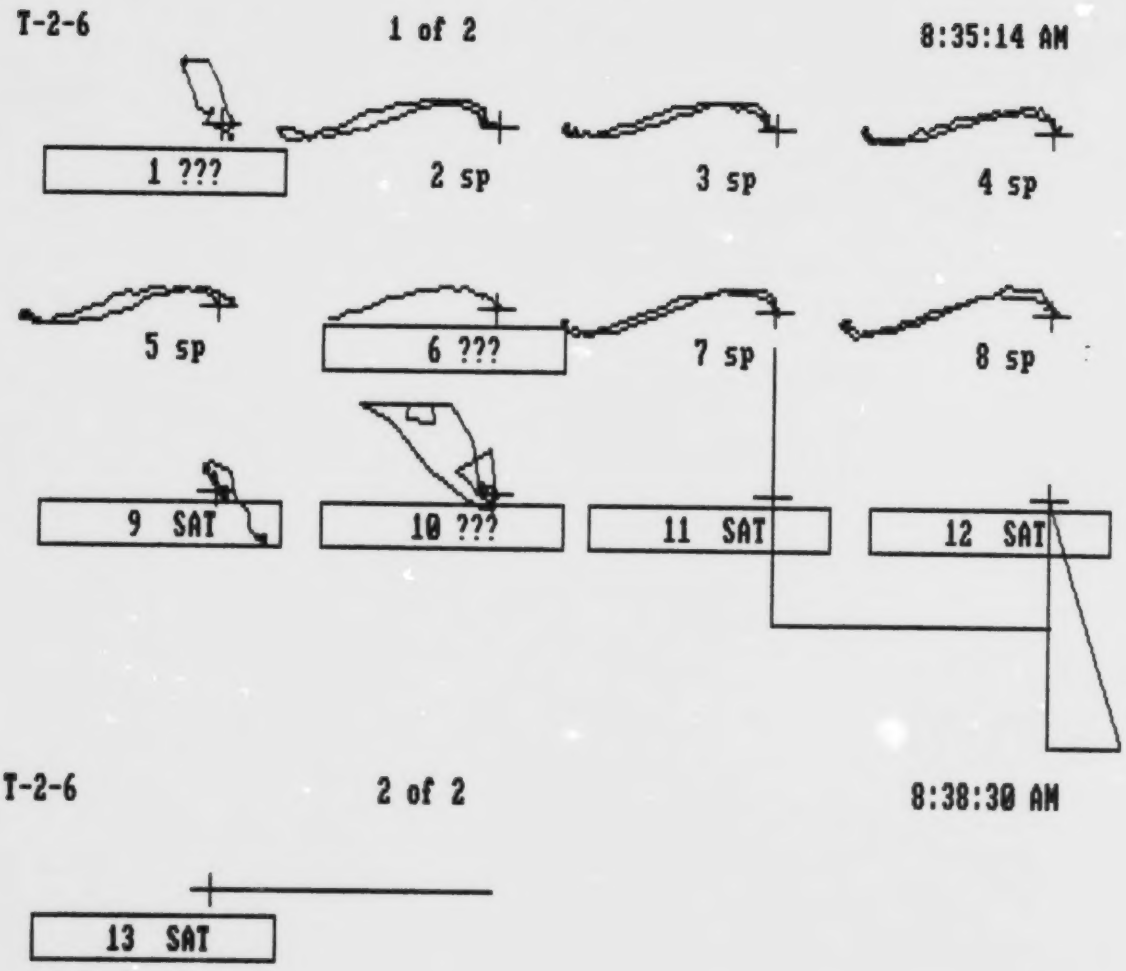


Figure 4.4 EXPECT results for T-2-6. The CNS analysis of this tube was: large amplitude OD + dent. This could be Event 1 which looks like an OD signal pulled along the -X axis but it is hard to tell because the signal is saturated.

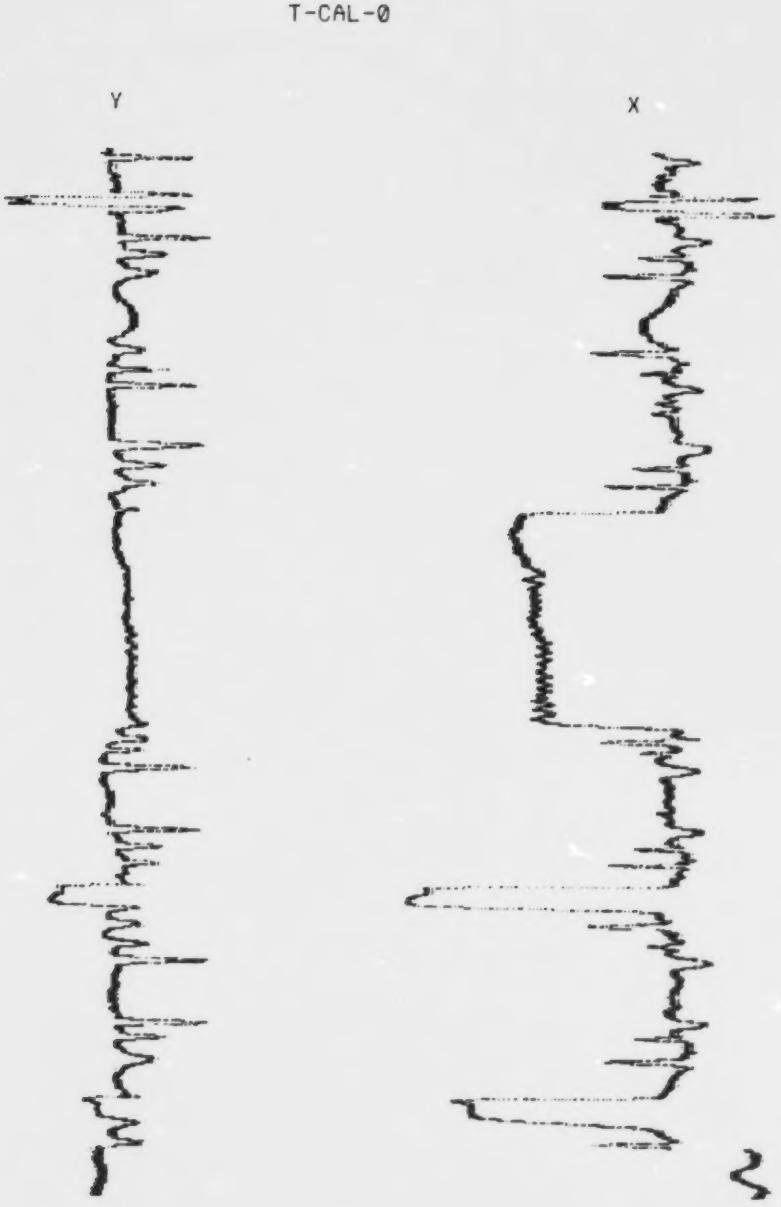


Figure 4.5 Separate X and Y components for tube T-CAL-0.

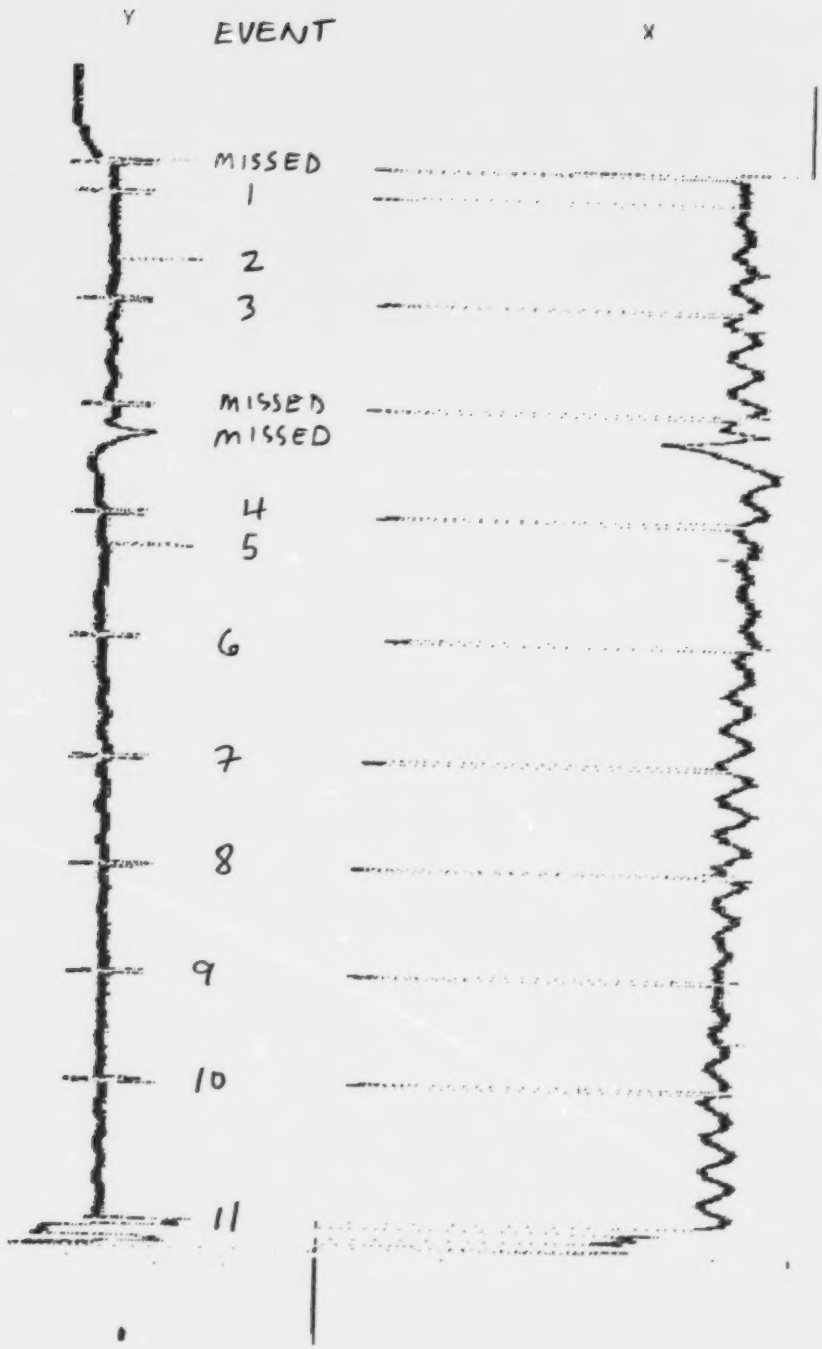


Figure 4.6 **Separate X and Y components for tube T-1-11. The components labelled with numbers correspond to the events of Figure 4.2. The peaks labelled "missed" are those that the feature extraction procedures of EXPECT did not pick out. The third missed signal from the top is probably due to a Whalie, since it looks to have the characteristic straight line shape and the signal is very broad compared to the support plate signals.**

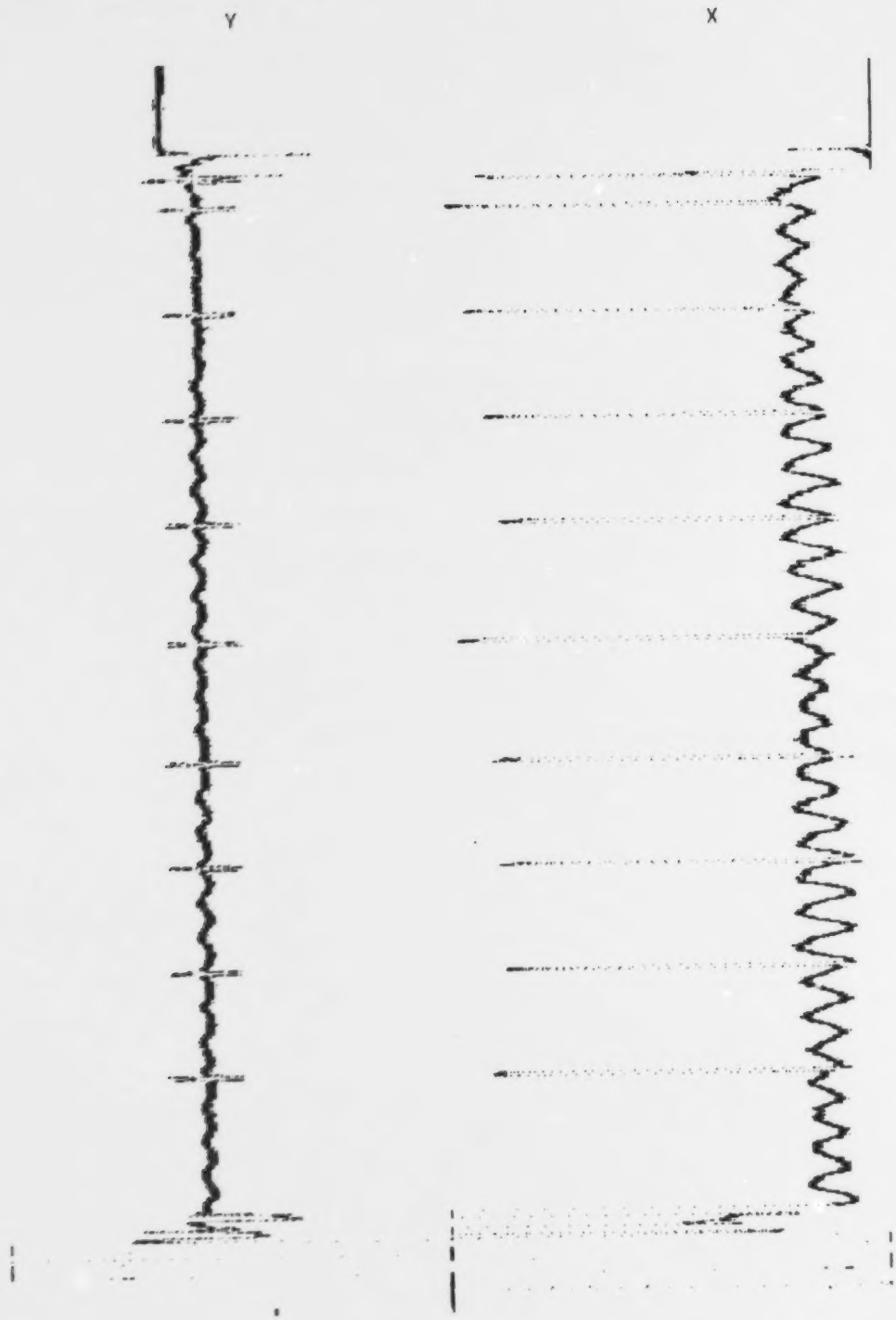


Figure 4.7 Separate X and Y components for tube T-1-22R.

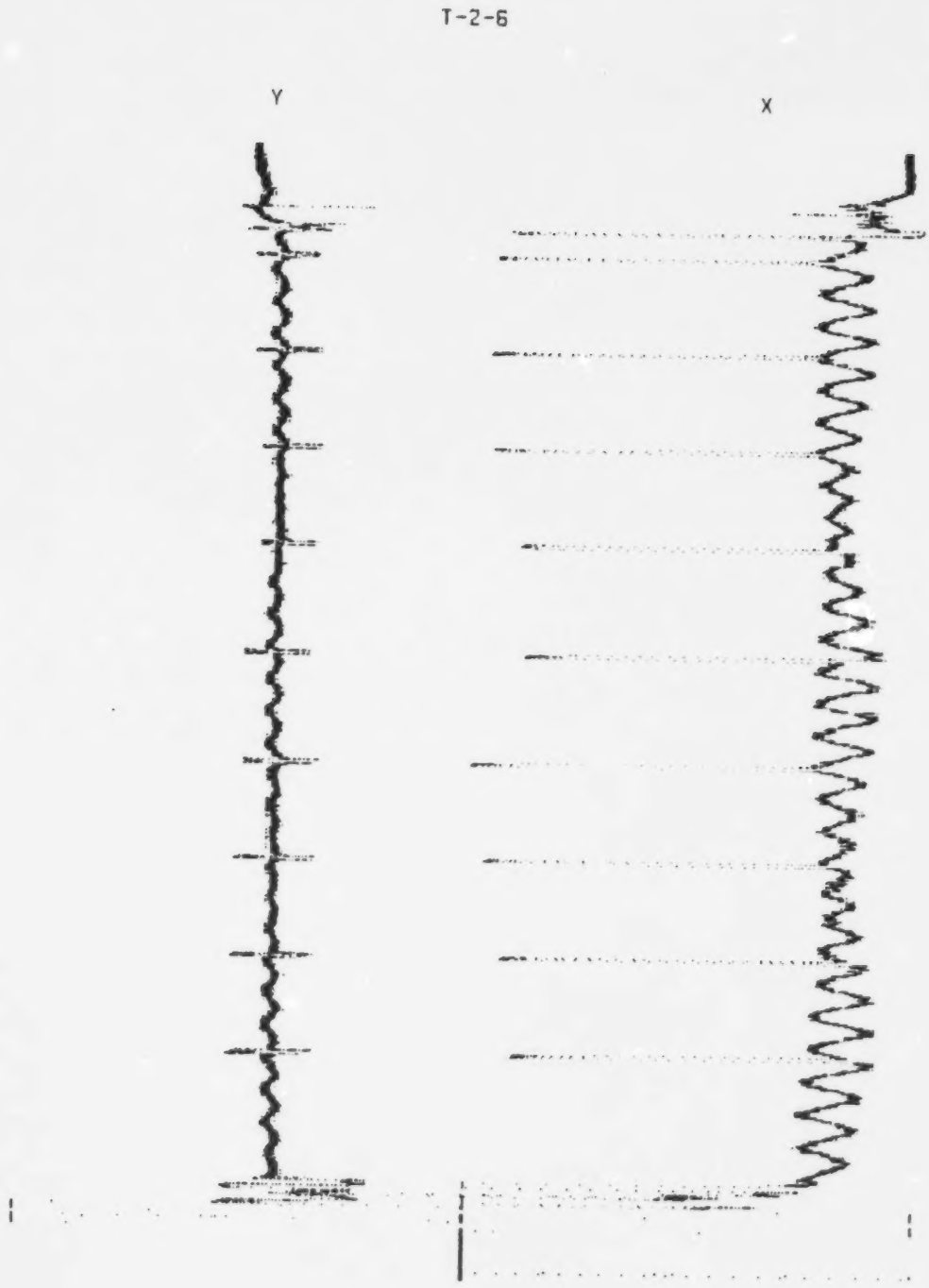


Figure 4.8 Separate X and Y components for tube T-2-6.

5. CONCLUSIONS AND RECOMMENDATIONS

Based on the work documented in this report, the following conclusions have been reached:

1. For isolated defects and normal signals the method used here (rules based on simple features) is very successful and can be made fast enough on an IBM AT or clone.

2. The software chosen for the implementing the prototype (Gold Hill's GCLISP) when augmented with C will be very satisfactory for delivering the mature expert system. Since GCLISP has a direct interface with C, and ANDES is written in C, EXPECT and ANDES can be easily integrated.

3. It has been demonstrated that any computer programmer who knows which features are calculated by EXPECT can add new rules (in LISP or C) without knowing anything of the EXPECT code or having to change EXPECT. This means that new rules can easily be added, independent of EXPECT's author. To go one step further and to endow EXPECT with a capability that would enable a non-programmer to add new rules has not been shown and would be a difficult (and interesting) challenge.

4. For composite signals (those made up of a normal signal and one or more defect signals or two or more defect signals, occurring at the same spot) it has not been demonstrated that the methods for analysis used here will be successful in practice. Five suggestions are made in Section 3.3.2 for methods that appear promising but need further investigation.

5. A potential problem with the analysis of composite signals is the non-uniqueness of the decomposition of signals into components. This will result in error limits in the analysis. More work is needed to quantify this.

6. The construction of a detailed knowledge base, such as started in Chapter 3 seems to be a worthwhile activity, independent of whether or not a practical on-line real time expert system is developed. It will promote standardization of analysis, be a vehicle for the interchange of experience among the analysts, be a method of capturing expertise so it will not be lost in the event of an analyst leaving, retiring etc, and can be used as an aid to training new analysts.

In order to continue this project it is necessary to have available a complete data base of isolated and composite signals. Since it is unlikely that a complete set of the necessary composite examples can be found from ANDES data it will be necessary to generate them from the isolated signals (such as was done for Figures 3.4 - 3.6 and 3.19).

Therefore, it is recommended that a complete data base of isolated signals be collected (including their normal statistical variation) and at the same time a method be developed that can be used to combine them to produce composite signals. As discussed in Section 3.3.2 this may be all that is required to satisfy the objectives of this project.

Following this, it is recommended that two projects be carried out in parallel:

1. The construction of a complete and systematic data base of the composite signals known to occur, initially for one type of tube and absolute probes, and then expanded to other types of tubes and finally to relative probes. Such a data base would be in the form of both descriptions and figures. It could be integrated with ANDES and available to the analyst, on-line.

2. Investigation of the problem of non-uniqueness in the decomposition of composite signals and the interrelated problem of the best way to analyze composite signals.

Following the successful completion of the above, an on-line EXPECT can be developed.

ACKNOWLEDGEMENTS

The author would like to thank John Sedo and Evan Choi, for reviewing the first draft of this report.

Approved:

for A.F. Baljet
Manager
Electrical Research Department

Submitted:

S.B. Nickerson
Theoretical Physicist
Science Section

APPENDIX A

EXPECT VARIABLES FOR TUBE T-1-11

```
FILE NAME  T-1-11
TUBE NUMBER  1 11
STATION
UNIT  6
HEAT EXCHANGER TYPE  Moderator Hx
HEAT EXCHANGER NUMBER  2
FREQUENCY  220k
DATE
TIME
SUPERVISOR  eg
COMMENT
AVERAGE TIP LENGTH  198.932
AVERAGE TIP ANGLE  -2.5
NORMAL SUPPORT PLATES  (1 3 4 6 7 8 9 10)

FILE NAME  T-1-11
EVENT  1
TIP-INDEX  65
TIP-ANGLE  -3
TIP-LENGTH  198.305
VECTOR-ENDS-OUT  (13 20 24 27 30 33 36 41)
VECTOR-ENDS-BACK  (113 108 104 101 98 95 92 87)
BASIS-OUT    (19 32 35 35 36 35 33 31)
             (- + + + + + + -)
BASIS-BACK   (25 30 32 34 37 35 35 35)
             (- - + + + + + +)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  107
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (1 4)
S-SHAPE-BACK? (2 4)
OUT=BACK?  T
PROSP?  T
NOSP?  T
ANALYSIS  sp



FILE NAME  T-1-11
EVENT  2
TIP-INDEX  28
TIP-ANGLE  98
TIP-LENGTH  44.4072
VECTOR-ENDS-OUT  (16 24)
VECTOR-ENDS-BACK  (41 32)
```

```
BASIS-OUT     (10 19)
              (- -)
BASIS-BACK    (10 20)
              (- -)
BASIS-LENGTH-OUT  2
BASIS-LENGTH-BACK 2
INITIAL-ANGLE-OUT  56
INITIAL-ANGLE-BACK 56
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  T
PROSP?  NIL
NOSP?  NIL
NN-VE-OUT  (0 12 14 18 21 26)
NN-VE-BACK  (62 49 45 41 38 37 33)
NN-B-OUT  (61 13 11 20 17 19)
NN-B-BACK  (58 8 12 16 24 15 18)
NN-BL-OUT  6
NN-BL-BACK  7
NN-IA-OUT  343
NN-IA-BACK  326
NN-O=B?  T
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  30
ANALYSIS   (OD - 30%  SAT)



FILE NAME  T-1-11
EVENT  3
TIP-INDEX  67
TIP-ANGLE  -3
TIP-LENGTH  199.203
VECTOR-ENDS-OUT  (14 20 23 26 29 32 35 39 46)
VECTOR-ENDS-BACK  (119 112 109 106 103 100 97 93 85)
BASIS-OUT     (21 31 34 35 36 35 34 34 29)
              (- - + + + + + + -)
BASIS-BACK    (26 29 33 33 35 36 35 36 31)
              (- - + + + + + + -)
BASIS-LENGTH-OUT  9
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  118
INITIAL-ANGLE-BACK 146
S-SHAPE-OUT?  (2 4)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
```

```
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-11
EVENT  4
TIP-INDEX  64
TIP-ANGLE  -3
TIP-LENGTH  199.203
VECTOR-ENDS-OUT  (16 20 23 26 29 32 35 40 61)
VECTOR-ENDS-BACK  (114 108 105 102 99 96 93 89 74)
BASIS-OUT    (22 32 35 35 36 35 33 32 28)
             (- + + + + + + + -)
BASIS-BACK   (24 31 32 34 35 36 35 34 28)
             (- - + + + + + + -)
BASIS-LENGTH-OUT  9
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  124
INITIAL-ANGLE-BACK 135
S-SHAPE-OUT?  (1 4)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-11
EVENT  5
TIP-INDEX  32
TIP-ANGLE  78
TIP-LENGTH  50.01
VECTOR-ENDS-OUT  (16 23)
VECTOR-ENDS-BACK  (48 39)
BASIS-OUT    (10 22)
             (- -)
BASIS-BACK   (16 21)
             (- -)
BASIS-LENGTH-OUT  2
BASIS-LENGTH-BACK 2
INITIAL-ANGLE-OUT  56
INITIAL-ANGLE-BACK 90
S-SHAPE-OUT?  F
```

```
S-SHAPE-BACK? F
OUT=BACK?  T
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (1 12 15 18 21 24 26 30)
NN-VE-BACK  (56 52 47 44 40 35)
NN-B-OUT  (63 13 18 16 22 18 26 27)
NN-B-BACK  (12 17 20 22 19 19)
NN-BL-OUT  8
NN-BL-BACK  6
NN-IA-OUT  354
NN-IA-BACK  68
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL



FILE NAME  T-1-11
EVENT  6
TIP-INDEX  72
TIP-ANGLE  -3
TIP-LENGTH  185.219
VECTOR-ENDS-OUT  (15 20 23 26 29 32 36 46)
VECTOR-ENDS-BACK  (127 122 119 116 113 110 106 100)
BASIS-OUT    (21 32 35 34 37 35 32 30)
             (- + + + + + + -)
BASIS-BACK   (25 30 33 34 35 36 35 31)
             (- - + + + + + -)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  118
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (1 4)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-11
EVENT  7
```

```
TIP-INDEX  78
TIP-ANGLE  -2
TIP-LENGTH  205.119
VECTOR-ENDS-OUT  (10 21 25 28 31 34 37 41 48)
VECTOR-ENDS-BACK  (136 130 125 122 1.9 116 113 108)
BASIS-OUT    (23 29 34 35 36 35 36 34 29)
             (- - + + + + + + -)
BASIS-BACK   (26 29 32 35 35 36 35 34)
             (- - + + + + + +)
BASIS-LENGTH-OUT  9
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  129
INITIAL-ANGLE-BACK 146
S-SHAPE-OUT?  (2 4)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-11
EVENT  8
TIP-INDEX  68
TIP-ANGLE  -2
TIP-LENGTH  204.12
VECTOR-ENDS-OUT  (15 22 26 29 32 35 39 45)
VECTOR-ENDS-BACK  (120 114 110 107 104 101 97 92 75)
BASIS-OUT    (22 30 34 35 36 35 34 32)
             (- - + + + + + +)
BASIS-BACK   (24 29 33 33 35 36 36 34 26)
             (- - + + + + + + -)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  124
INITIAL-ANGLE-BACK 135
S-SHAPE-OUT?  (2 4)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp
```

```
FILE NAME  T-1-11
EVENT  9
TIP-INDEX  68
TIP-ANGLE  -2
TIP-LENGTH  203.158
VECTOR-ENDS-OUT  (14 21 25 28 31 34 38 44)
VECTOR-ENDS-BACK  (121 115 111 107 104 101 98 94 85)
BASIS-OUT    (22 31 34 36 36 35 34 31)
             (- - + + + + + -)
BASIS-BACK   (25 30 31 34 35 36 36 35 30)
             (- - - + + + + + -)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  124
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (2 3)
S-SHAPE-BACK? (3 5)
OUT=BACK?  T
PROSP?  T
NOSP?  T
ANALYSIS  sp


FILE NAME  T-1-11
EVENT  10
TIP-INDEX  73
TIP-ANGLE  -2
TIP-LENGTH  197.124
VECTOR-ENDS-OUT  (15 21 25 28 31 34 37 42)
VECTOR-ENDS-BACK  (133 127 123 120 117 114 111 108 101)
BASIS-OUT    (22 32 34 36 36 36 33 31)
             (- + + + + + + -)
BASIS-BACK   (24 30 32 34 35 36 36 36 32)
             (- - + + + + + + +)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  124
INITIAL-ANGLE-BACK 135
S-SHAPE-OUT?  (1 3)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
PROSP?  T
NOSP?  T
ANALYSIS  sp
```

```
FILE NAME  T-1-11
EVENT  11
TIP-INDEX  66
TIP-ANGLE  19
TIP-LENGTH  171.735
VECTOR-ENDS-OUT  (0)
VECTOR-ENDS-BACK  (156 152 149 146 142 136 132 117)
BASIS-OUT    (29)
             (-)
BASIS-BACK   (26 30 31 31 30 31 29 18)
             (- - - - - - - -)
BASIS-LENGTH-OUT  1
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  163
INITIAL-ANGLE-BACK 146
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?  NIL
NN-VE-OUT  (0)
NN-VE-BACK  (166 159 156 154 153 152 151 149 148 147 145 144 142 139 137 135 134 133 1
NN-B-OUT  (29)
NN-B-BACK  (17 28 30 28 28 33 29 31 31 30 30 31 30 30 31 32 29 32 25 22 16 16)
NN-BL-OUT  1
NN-BL-BACK  22
NN-IA-OUT  163
NN-IA-BACK  96
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  NIL
WALL-LOSS  NIL
ANALYSIS   ( SAT)
```

APPENDIX B

DATA STRUCTURE FOR TUBE T-1-11

```
(setf T-1-11#s
(TUBE :SLOT-LIST (PARAMETERS INFO EVENT LIMITS CALCULATED-PROPERTIES NOSP NON-
NOSP ANALYSIS) :PARAMETERS #s(PARAMETERS :OR-TRIGGER 5000 :AND-TRIGGER 250 :OR-M
INIMUM 25 :AND-MINIMUM 5 :AND-MINIMUM-LENGTH 5 :ANGLES 64 :TIP 0.9 :BASIS-LENGTH
 0.1 :LENGTH-COMP 0.2 :DIFF-VECT 1 :NOSP-LENGTH 0.1 :NOSP-ANGLE 5 :NON-N-BASIS-L
ENGTH 9) :INFO #s(INFO :TUBE-NUMBER "1 11" :STATION "" :UNIT "6" :HEAT-EXCHANGER
-TYPE "Moderator Hx" :HEAT-EXCHANGER-NUMBER "2" :FREQUENCY "220k" :DATE "" :TIME
 "" :SUPERVISOR "eg" :COMMENT "") :EVENT #s(EVENT :X ((2 2 2 3 1 3 2 3 1 1 -1 -2
 -3 -6 -7 -12 -14 -17 -23 -27 -34 -39 -46 -54 -60 -70 -79 -85 -93 -102 -112 -122
 -129 -137 -145 -151 -158 -164 -169 -173 -177 -180 -182 -187 -188 -189 -190 -190
 -193 -194 -193 -192 -194 -194 -193 -194 -196 -194 -194 -196 -197 -196 -198 -194
 -197 -198 -199 -196 -198 -197 -200 -198 -197 -198 -198 -200 -199 -198 -199 -196
 -199 -199 -196 -194 -194 -192 -188 -185 -182 -180 -175 -169 -163 -156 -150 -142
 -134 -128 -117 -110 -100 -94 -84 -74 -66 -58 -52 -46 -41 -34 -30 -26 -21 -18 -1
4 -10 -10 -8 -6 -2 -2 -5 -2 -2 -2 -2 -2 2 -2 -2 0 2 2 0 2 2 1 1 1 0) (7 6 6 7 8
8 10 10 10 11 10 9 10 14 14 12 13 14 11 10 12 10 11 8 6 8 8 6 6 6 7 6 6 8 7 9 10
 10 9 14 11 14 14 14 14 14 14 14 12 11 12 10 10 6 9 10 6 6 6 6 6 4 6 4 3 3 6 4 4
 4 4 4 3 4 5 4 4) (-1 1 1 1 0 1 1 -1 -1 -3 -2 -6 -5 -7 -11 -13 -16 -21 -23 -30 -
33 -39 -47 -55 -59 -69 -77 -83 -91 -101 -107 -117 -124 -133 -140 -148 -154 -161
-164 -170 -175 -180 -183 -185 -188 -190 -191 -192 -192 -193 -196 -195 -195 -198
-197 -195 -198 -197 -198 -197 -198 -198 -197 -198 -199 -197 -199 -199 -197 -199
-199 -199 -199 -199 -199 -200 -200 -202 -200 -199 -201 -202 -201 -198 -199 -200
-199 -198 -194 -193 -190 -185 -182 -179 -173 -167 -162 -155 -147 -142 -132 -126
-117 -109 -99 -93 -84 -77 -69 -63 -55 -49 -41 -39 -34 -30 -26 -23 -21 -19 -15 -1
4 -13 -12 -7 -7 -9 -7 -7 -7 -7 -6 -3 -6 -7 -7 -5 -7 -7 -7 -5 -3 -5 -5 -6 -6 -5 -
6 -7 -6 -6 -6 -3 -6 -7 -7 -7) (9 8 8 8 8 8 8 6 6 4 4 0 0 -3 -6 -10 -12 -18 -24 -
30 -36 -43 -52 -58 -64 -76 -82 -92 -101 -109 -116 -126 -135 -140 -149 -156 -160
-166 -171 -176 -178 -180 -184 -187 -188 -189 -188 -192 -192 -193 -195 -192 -194
-195 -196 -195 -197 -197 -195 -197 -197 -198 -196 -197 -199 -196 -197 -197 -197
-198 -196 -197 -198 -198 -198 -197 -196 -194 -196 -194 -196 -195 -193 -195 -192
-192 -187 -187 -183 -177 -173 -167 -163 -156 -148 -143 -133 -127 -119 -109 -103
-94 -86 -79 -70 -64 -56 -49 -43 -36 -33 -28 -24 -20 -16 -14 -12 -10 -8 -7 -6 -5
-4 -3 -2 -2 -2 -1 0 0 0 0 0 0 0 0 0 0 0 0 0) (6 8 8 4 6 6 7 6 7 7 8 8 10 8 9 8 12
8 8 8 7 4 2 1 2 -2 -4 -4 -8 -9 -11 -8 -10 -10 -11 -11 -11 -10 -12 -10 -9 -6 -7 -
7 -6 -4 -2 -2 -1 0 0 1 2 1 3 3 3 4 3 4 3 4 2 2 2 2 2 0 0 1 2 0 0 1 2 1) (11 8 8
8 7 6 7 8 7 4 3 2 -1 -1 -5 -11 -15 -18 -25 -30 -37 -45 -53 -61 -69 -77 -87 -93 -
101 -111 -120 -126 -135 -141 -148 -153 -159 -161 -167 -169 -173 -175 -177 -177 -
179 -179 -181 -181 -178 -181 -181 -181 -180 -181 -181 -181 -179 -181 -179 -179 -
181 -182 -181 -181 -184 -181 -183 -185 -186 -185 -185 -183 -185 -187 -185 -187 -
185 -185 -185 -189 -188 -188 -189 -187 -189 -188 -189 -191 -191 -187 -191 -190 -
190 -190 -190 -189 -190 -189 -189 -188 -186 -185 -180 -180 -177 -171 -165 -161 -
155 -148 -141 -133 -125 -116 -108 -100 -93 -81 -72 -65 -57 -47 -42 -33 -30 -26 -
19 -16 -13 -8 -5 -2 -1 0 1 1 2 4 4 6 5 7 7 5 5 7 3 5 5 5 7 6 5 4) (-10 -13 -11 -
12 -13 -13 -13 -11 -13 -13 -13 -13 -14 -15 -18 -19 -21 -22 -25 -28 -32 -38 -42 -
48 -53 -60 -68 -73 -81 -86 -95 -103 -110 -118 -126 -134 -142 -148 -152 -160 -166
 -173 -177 -181 -185 -186 -190 -190 -194 -195 -197 -196 -199 -198 -198 -199 -198
 -198 -200 -199 -202 -200 -201 -201 -201 -201 -202 -200 -202 -203 -202 -203 -202
 -203 -204 -204 -205 -204 -205 -202 -206 -203 -202 -202 -206 -205 -206 -206 -207
```

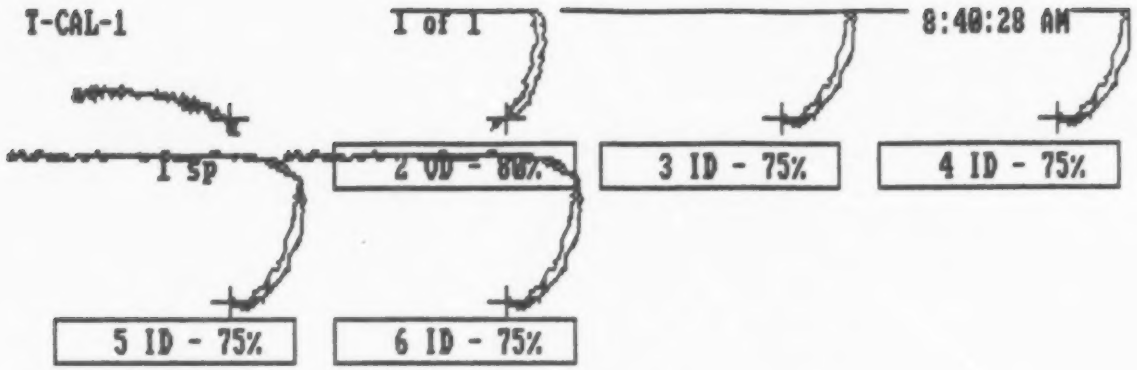
-206 -206 -207 -205 -202 -206 -205 -205 -202 -204 -204 -205 -203 -204 -201 -198
-199 -197 -196 -192 -186 -184 -178 -173 -168 -160 -153 -146 -137 -130 -121 -113
-106 -97 -88 -80 -72 -62 -58 -51 -46 -42 -37 -33 -28 -26 -22 -19 -16 -14 -14 -1
4 -11 -10 -10 -9 -9 -8 -8 -6 -6 -6 -6 -7 -5 -5 -6 -5 -4 -4 -5 -5 -5 -5 -5 -3 -4
-6 -4 -5 -4 -3 -4 -6) (-5 -5 -7 -5 -5 -5 -5 -5 -6 -6 -5 -6 -8 -10 -12 -12 -13 -1
6 -17 -21 -27 -31 -37 -42 -48 -53 -61 -71 -77 -85 -95 -102 -112 -121 -129 -138 -
145 -154 -158 -167 -171 -176 -181 -185 -187 -191 -194 -195 -197 -198 -200 -200 -
197 -202 -204 -202 -203 -203 -203 -202 -203 -201 -205 -204 -203 -205 -205 -206 -
204 -205 -205 -205 -204 -202 -205 -206 -205 -204 -204 -205 -204 -203 -201 -203 -
201 -201 -200 -198 -197 -197 -192 -191 -187 -184 -179 -173 -169 -164 -155 -150 -
142 -137 -127 -120 -114 -105 -96 -87 -82 -74 -65 -57 -51 -44 -37 -33 -29 -25 -21
-17 -15 -13 -12 -8 -7 -7 -5 -3 -4 -6 -4 -4 -4 -2 1 -1 -1 -2 0 1 -1 -1 -2 -1 -1
-1 -1 -1 -1 -1 -1) (-7 -5 -6 -6 -8 -4 -6 -7 -8 -7 -7 -8 -12 -12 -13 -16 -16 -19
-24 -30 -32 -40 -45 -53 -60 -70 -76 -84 -94 -104 -112 -121 -128 -136 -143 -148 -
154 -160 -169 -172 -176 -180 -184 -187 -190 -193 -192 -194 -195 -196 -198 -198 -
200 -200 -201 -201 -198 -200 -201 -201 -203 -201 -200 -200 -203 -200 -202 -205 -
203 -203 -205 -206 -204 -205 -203 -203 -203 -200 -203 -203 -205 -204 -204 -203 -
202 -201 -200 -199 -198 -196 -194 -191 -187 -180 -180 -174 -168 -163 -158 -151 -
143 -135 -129 -120 -112 -105 -98 -90 -80 -74 -67 -61 -52 -46 -40 -37 -32 -28 -24
-22 -20 -16 -13 -13 -12 -10 -10 -8 -11 -4 -8 -6 -7 -5 -6 -6 -4 -6 -4 -4 -5 -2 -
4 -4 -7 -4 -4 -4 -4 -4 -4 -4) (0 0 0 1 0 0 0 -2 0 -4 -4 -6 -8 -8 -10 -14 -17 -21
-23 -29 -32 -38 -43 -48 -54 -64 -72 -76 -88 -98 -106 -114 -124 -131 -138 -146 -
154 -163 -167 -171 -176 -181 -184 -186 -188 -189 -188 -192 -191 -192 -192 -193 -
192 -194 -193 -192 -193 -194 -194 -195 -192 -196 -197 -195 -195 -197 -194 -197 -
196 -196 -196 -198 -197 -197 -199 -200 -198 -197 -199 -199 -198 -200 -200 -200 -
199 -199 -201 -199 -200 -200 -200 -201 -200 -200 -200 -200 -199 -198 -196 -198 -
198 -194 -192 -192 -188 -185 -180 -175 -171 -164 -158 -151 -144 -137 -131 -123 -
116 -108 -100 -91 -84 -79 -71 -63 -57 -52 -45 -40 -35 -30 -26 -24 -20 -16 -12 -1
2 -10 -8 -7 -7 -6 -5 -6 -4 -4 -4 -4 -4 -4 -2 -4 -2 -3 -5 -2 -4 -4 -5 -3 -5 0) (-
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -162 -
162 -162 -162 -162 -158 -151 -143 -136 -132 -127 -122 -118 -114 -110 -108 -100 -
96 -89 -83 -78 -70 -62 -54 -48 -41 -34 -26 -22 -18 -13 -10 -8 -7 -4 -2 -2 -1 -1
-1 0 2 5 6 3 4 6 6 2 2 6 4 3 2 0 2)) :X-BASE (90 95 88 88 93 98 93 92 88 83 20)
:Y ((5 6 8 5 5 7 9 9 11 13 13 17 17 21 18 20 20 21 21 21 20 20 17 15 13 10 9 5 1
-2 -7 -8 -12 -15 -15 -17 -18 -17 -17 -15 -15 -15 -12 -13 -11 -12 -11 -11 -9 -9
-10 -9 -11 -10 -9 -11 -10 -10 -11 -10 -11 -12 -11 -9 -11 -11 -12 -11 -11 -11 -11
-11 -11 -11 -11 -11 -12 -13 -13 -15 -15 -16 -16 -18 -18 -19 -19 -21 -19 -18 -20
-17 -14 -14 -11 -7 -4 1 1 4 9 12 14 16 19 19 19 19 19 17 18 16 14 14 14 11 11 1
1 9 8 6 7 7 5 3 5 2 2 2 4 1 1 1 1 5 1 1 1 1 1) (-2 -3 -2 -4 -2 -1 -1 -1 3 3 3 3
7 11 15 15 19 21 23 26 29 35 35 38 39 40 42 44 44 44 45 41 40 39 35 35 33 31 24
23 19 19 15 12 10 8 7 4 3 1 -1 -1 -1 -3 -2 -1 -3 -2 -2 -2 -2 -4 -4 -3 -3 -3 -2 -

4 -4 -2 -2 -1 -2 -4 -3 -3 0) (4 4 5 5 7 9 9 10 13 13 13 13 15 18 19 19 21 21 20
21 21 20 20 17 16 13 11 6 5 1 -2 -5 -7 -10 -13 -11 -13 -15 -15 -15 -13 -11 -13 -
11 -11 -11 -9 -9 -9 -7 -7 -7 -7 -7 -7 -7 -9 -7 -9 -7 -9 -7 -9 -9 -9 -11 -10 -9 -
9 -11 -10 -11 -10 -9 -11 -9 -11 -9 -11 -9 -11 -12 -12 -12 -14 -15 -15 -15 -15 -1
7 -18 -18 -19 -18 -17 -15 -13 -9 -7 -5 -1 3 5 9 11 13 17 17 20 19 19 20 21 19 19
17 17 14 15 14 12 10 10 9 9 9 6 7 6 6 6 4 5 3 4 3 3 4 2 2 1 2 3 5 3 3 1 1 5 1 1
1 1 1 2 5 1) (4 5 5 5 7 8 8 9 12 14 14 15 17 17 17 18 19 19 21 19 18 19 15 12 1
0 8 5 3 -1 -5 -8 -12 -13 -14 -15 -15 -16 -17 -15 -16 -15 -13 -14 -13 -13 -9 -11
-11 -9 -9 -8 -8 -8 -9 -7 -10 -9 -8 -9 -9 -9 -7 -9 -9 -9 -11 -9 -9 -9 -8 -9 -9 -9
-9 -9 -11 -11 -11 -11 -13 -13 -13 -13 -17 -16 -17 -17 -17 -17 -17 -17 -17 -15 -
13 -9 -7 -5 -1 3 6 7 10 13 15 17 18 19 19 19 19 19 19 17 15 15 12 13 10 8 8 7 6
6 5 3 4 2 3 3 2 3 2 3 0 0 0 0 0 0 0 0) (0 -1 0 0 1 1 1 3 1 2 5 5 6 8 12 14 18 19 2
5 26 28 31 35 36 40 44 44 46 47 48 48 48 49 48 48 47 46 44 43 41 40 36 36 36 31
28 26 25 22 21 17 16 16 12 10 9 8 5 5 5 4 3 1 1 3 3 2 1 1 -1 3 2 2 4 2 3) (6 4 4
6 8 8 10 11 11 11 13 16 18 18 20 19 20 20 19 18 18 15 14 10 6 5 5 -2 -5 -7 -10
-10 -14 -14 -14 -14 -15 -17 -15 -15 -15 -13 -10 -11 -13 -10 -11 -10 -9 -10 -9 -1
0 -8 -6 -8 -8 -8 -9 -8 -8 -6 -8 -8 -6 -8 -8 -10 -11 -8 -8 -9 -10 -9 -10 -9 -9 -1
0 -9 -9 -9 -10 -9 -8 -8 -10 -9 -8 -10 -9 -10 -9 -10 -10 -10 -10 -12 -12 -12 -12
-14 -14 -17 -16 -16 -16 -14 -16 -14 -12 -11 -9 -5 -4 2 2 6 10 12 13 15 18 16 18
18 16 18 18 14 15 14 12 12 9 8 9 8 6 6 4 4 4 5 3 3 3 2 4 2 3 4 1 2 0 2) (7 7 7 6
9 9 9 11 13 13 17 17 18 17 19 21 21 23 25 25 25 25 25 23 23 21 21 17 14 11 9 5
2 -2 -3 -6 -8 -11 -15 -14 -15 -15 -13 -13 -12 -12 -11 -10 -9 -9 -7 -7 -6 -6 -6 -
7 -3 -6 -6 -7 -6 -5 -7 -5 -5 -7 -5 -4 -5 -7 -4 -7 -3 -7 -5 -6 -7 -7 -7 -6 -6 -7
-7 -7 -6 -7 -5 -7 -7 -5 -7 -7 -7 -7 -6 -7 -8 -7 -7 -7 -8 -11 -10 -11 -12 -13 -15
-15 -16 -17 -15 -15 -15 -11 -10 -7 -5 -1 1 5 9 11 13 15 20 20 20 22 21 22 20 19
19 18 17 16 13 13 12 10 10 9 9 9 7 7 5 6 5 4 5 4 5 5 3 5 2 2 2 2 4 3 5 2 1 4 2
1 2 3 1 1 4) (6 5 4 7 9 9 9 8 9 9 11 13 15 15 17 19 19 21 23 27 25 24 24 23 21 2
1 18 15 14 11 7 3 -1 -3 -7 -9 -10 -13 -13 -15 -17 -13 -16 -15 -13 -14 -13 -13 -1
0 -9 -11 -9 -9 -8 -9 -8 -8 -6 -7 -5 -5 -7 -7 -7 -8 -7 -7 -5 -7 -8 -9 -5 -7 -7 -7
-5 -7 -5 -9 -8 -9 -7 -9 -11 -12 -13 -12 -13 -14 -15 -17 -16 -17 -17 -14 -15 -11
-13 -9 -7 -5 -2 2 5 7 10 13 16 16 19 19 19 21 21 21 19 19 21 16 15 15 16 13 11
9 10 9 7 7 7 5 6 5 5 5 4 3 3 4 3 4 3 3 3 3 3 3 3 3 3) (4 4 6 6 7 6 7 10 10 10
14 13 14 15 18 20 20 21 22 22 22 22 20 18 16 15 13 10 5 1 -2 -6 -8 -10 -12 -13 -
14 -17 -18 -17 -14 -17 -16 -16 -15 -14 -14 -11 -10 -10 -9 -10 -8 -9 -8 -6 -8 -9
-8 -8 -8 -6 -8 -7 -9 -8 -8 -6 -8 -6 -8 -9 -8 -6 -8 -6 -6 -8 -8 -10 -6 -9 -10 -10
-10 -12 -10 -14 -14 -16 -16 -15 -18 -17 -16 -14 -15 -12 -10 -7 -5 -1 1 4 8 10 1
1 14 18 18 19 20 22 22 20 18 17 17 15 15 14 14 12 11 9 9 8 10 7 6 6 4 4 4 3 5 3
6 3 3 2 2 3 4 2 2 2 5 1 2 4 2) (5 4 6 7 7 7 11 10 13 13 16 14 16 17 17 20 19 21
21 19 21 19 19 18 15 13 10 10 2 0 -4 -8 -11 -13 -17 -18 -19 -19 -19 -19 -18 -18
-17 -17 -13 -15 -14 -11 -11 -11 -11 -11 -9 -11 -11 -7 -11 -8 -9 -7 -11 -8 -9 -7
-7 -9 -9 -11 -7 -7 -11 -8 -9 -7 -9 -10 -9 -9 -9 -7 -9 -10 -11 -8 -9 -9 -7 -9 -11
-8 -9 -11 -11 -11 -10 -11 -13 -13 -17 -14 -17 -18 -19 -19 -19 -19 -18 -17 -19 -
15 -13 -11 -9 -5 -3 1 5 7 8 13 15 17 18 19 21 19 21 19 19 19 17 19 15 15 13 12 1
1 11 9 7 5 6 5 5 5 3 4 3 3 3 4 3 3 3 1 3 2 2 1 3 1) (53 53 53 53 53 53 53 55 57
55 57 57 56 57 57 57 58 57 57 58 57 57 57 57 57 57 57 57 57 58 59 57 58 57 57 57
57 58 57 57 57 55 58 57 57 57 57 55 57 55 57 57 57 57 57 55 57 58 56 56 57 53 5
7 57 57 57 57 55 57 55 57 55 57 57 57 57 57 55 57 57 57 59 58 58 59 58 58 57 59
58 58 61 61 59 61 61 61 59 60 61 61 61 61 63 61 61 61 61 63 61 61 61 61 59 60 61

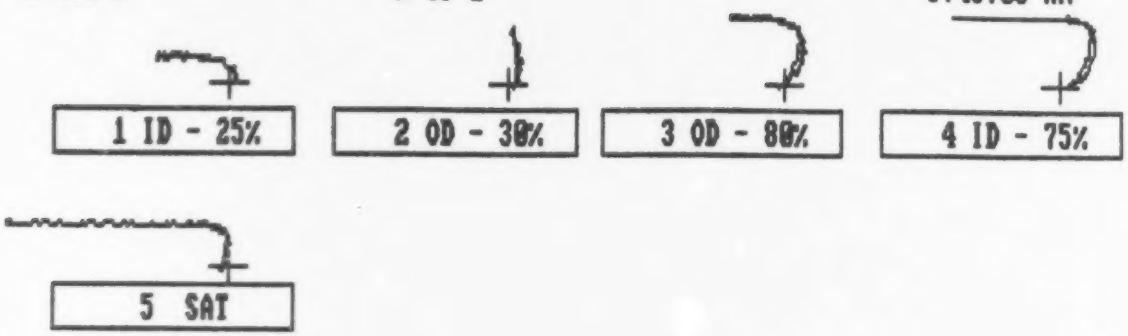
59 60 59 59 57 57 59 55 53 51 49 47 45 43 41 39 39 33 33 31 30 31 29 30 29 27 2
8 25 26 25 23 23 21 20 18 19 17 18 15 13 12 11 13 10 9 9 9 9 9 5 7 4 5 3 3 1 2 1
-2 -1 -1 -1 -2 -3 -3 -7 -5)) :Y-BASE (9 7 9 10 6 7 9 10 11 12 -25)) :LIMITS #s(
LIMITS :MAX-X 14 :MIN-X -207 :MAX-Y 63 :MIN-Y -21 :MAX-Y-MIN-X NIL) :CALCULATED-
PROPERTIES #s(CALC :TIP-INDEX (65 28 67 64 32 72 78 68 68 73 66) :TIP-ANGLE (-3
98 -3 -3 78 -3 -2 -2 -2 -2 19) :TIP-LENGTH (198.305 44.4072 199.203 199.203 50.0
1 185.219 205.119 204.12 203.158 197.124 171.735) :VECTOR-ENDS-OUT ((13 20 24 27
30 33 36 41) (16 24) (14 20 23 26 29 32 35 39 46) (16 20 23 26 29 32 35 40 61)
(16 23) (15 20 23 26 29 32 36 46) (10 21 25 28 31 34 37 41 48) (15 22 26 29 32 3
5 39 45) (14 21 25 28 31 34 38 44) (15 21 25 28 31 34 37 42) (0)) :VECTOR-ENDS-B
ACK ((113 108 104 101 98 95 92 87) (41 32) (119 112 109 106 103 100 97 93 85) (1
14 108 105 102 99 96 93 89 74) (48 39) (127 122 119 116 113 110 106 100) (136 13
0 125 122 119 116 113 108) (120 114 110 107 104 101 97 92 75) (121 115 111 107 1
04 101 98 94 85) (133 127 123 120 117 114 111 108 101) (156 152 149 146 142 136
132 117)) :BASIS-OUT ((19 32 35 35 36 35 33 31) (10 19) (21 31 34 35 36 35 34 34
29) (22 32 35 35 36 35 33 32 28) (10 22) (21 32 35 34 37 35 32 30) (23 29 34 35
36 35 36 34 29) (22 30 34 35 36 35 34 32) (22 31 34 36 36 35 34 31) (22 32 34 3
6 36 36 33 31) (29)) :BASIS-BACK ((25 30 32 34 37 35 35 35) (10 20) (26 29 33 33
35 36 35 36 31) (24 31 32 34 35 36 35 34 28) (16 21) (25 30 33 34 35 36 35 31)
(26 29 32 35 35 36 35 34) (24 29 33 33 35 36 36 34 26) (25 30 31 34 35 36 36 35
30) (24 30 32 34 35 36 36 36 32) (26 30 31 31 30 31 29 18)) :BASIS-LENGTH-OUT (8
2 9 9 2 8 9 8 8 8 1) :BASIS-LENGTH-BACK (8 2 9 9 2 8 8 9 9 9 8) :INITIAL-ANGLE-
OUT (107 56 118 124 56 118 129 124 124 124 163) :INITIAL-ANGLE-BACK (141 56 146
135 90 141 146 135 141 135 146)) :NOSP #s(NOSP :S-SHAPE-OUT? ((1 4) F (2 4) (1 4
) F (1 4) (2 4) (2 4) (2 3) (1 3) F) :S-SHAPE-BACK? ((2 4) F (2 5) (2 5) F (2 5)
(2 5) (2 5) (3 5) (2 5) F) :OUT=BACK? (T T T T T T T T T T F) :PROSP? (T NIL T
T NIL T T T T T NIL) :PROSP-AVG-TIP-LENGTH 198.932 :PROSP-AVG-TIP-ANGLE -2.5 :NO
SP? (T NIL T T NIL T T T T T NIL) :NOSP-EVENTS (0 2 3 5 6 7 8 9) :NOSP-AVG-TIP-L
ENGTH 198.932 :NOSP-AVG-TIP-ANGLE -2.5) :NON-NOSP #s(NON-NOSP :NN-VE-OUT (NIL (0
12 14 18 21 26) NIL NIL (1 12 15 18 21 24 26 30) NIL NIL NIL NIL NIL (0)) :NN-V
E-BACK (NIL (62 49 45 41 38 37 33) NIL NIL (56 52 47 44 40 35) NIL NIL NIL NIL N
IL (166 159 156 154 153 152 151 149 148 147 145 144 142 139 137 135 134 133 132
128 124 115)) :NN-B-OUT (NIL (61 13 11 20 17 19) NIL NIL (63 13 18 16 22 18 26 2
7) NIL NIL NIL NIL NIL (29)) :NN-B-BACK (NIL (58 8 12 16 24 15 18) NIL NIL (12 1
7 20 22 19 19) NIL NIL NIL NIL NIL (17 28 30 28 28 33 29 31 31 30 30 31 30 30 31
32 29 32 25 22 16 16)) :NN-BL-OUT (NIL 6 NIL NIL 8 NIL NIL NIL NIL NIL 1) :NN-B
L-BACK (NIL 7 NIL NIL 6 NIL NIL NIL NIL NIL 22) :NN-IA-OUT (NIL 343 NIL NIL 354
NIL NIL NIL NIL NIL 163) :NN-IA-BACK (NIL 326 NIL NIL 68 NIL NIL NIL NIL NIL 96)
:NN-O=B? (NIL T NIL NIL F NIL NIL NIL NIL NIL F) :CURVE-TO--X-OUT? (NIL T NIL N
IL T NIL NIL NIL NIL NIL NIL) :CURVE-TO--X-BACK? (NIL T NIL NIL T NIL NIL NIL NI
L NIL NIL) :WALL-LOSS (NIL 30 NIL NIL NIL NIL NIL NIL NIL NIL NIL)) :ANALYSIS #s
(ANALYSIS :LIST ("sp" ("OD - 30%" " SAT") "sp" "sp" NIL "sp" "sp" "sp" "sp" "sp"
(" SAT")))))

APPENDIX C

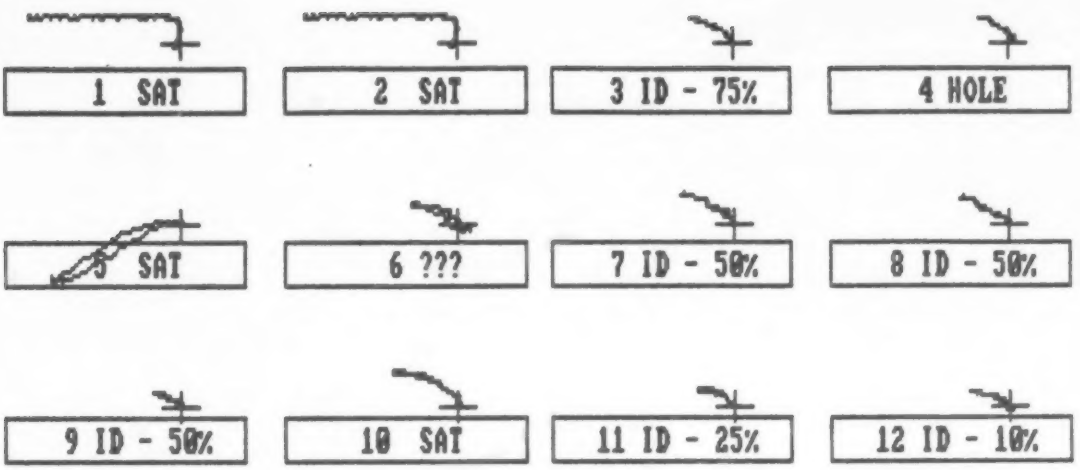
USING EXPECT WITH TUBES OTHER THAN TUBE T-1-11


T-CAL-1
1 of 1
8:40:28 AM
1 SP
2 OD - 80%
3 ID - 75%
4 ID - 75%
5 ID - 75%
6 ID - 75%

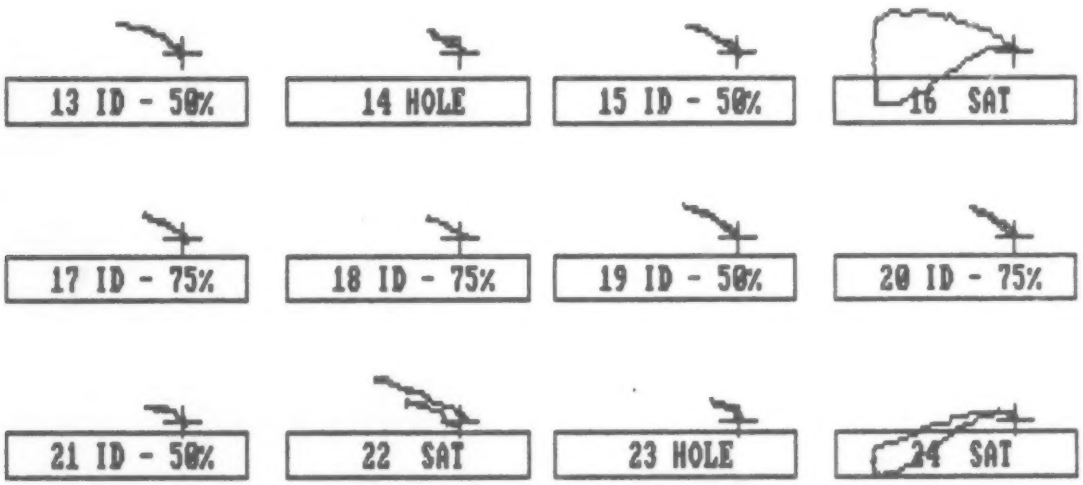
T-CAL-2 1 of 1 8:43:03 AM

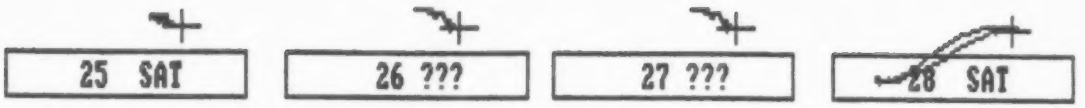


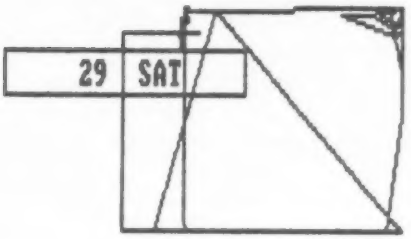
T-13-39 1 of 3 8:45:45 AM



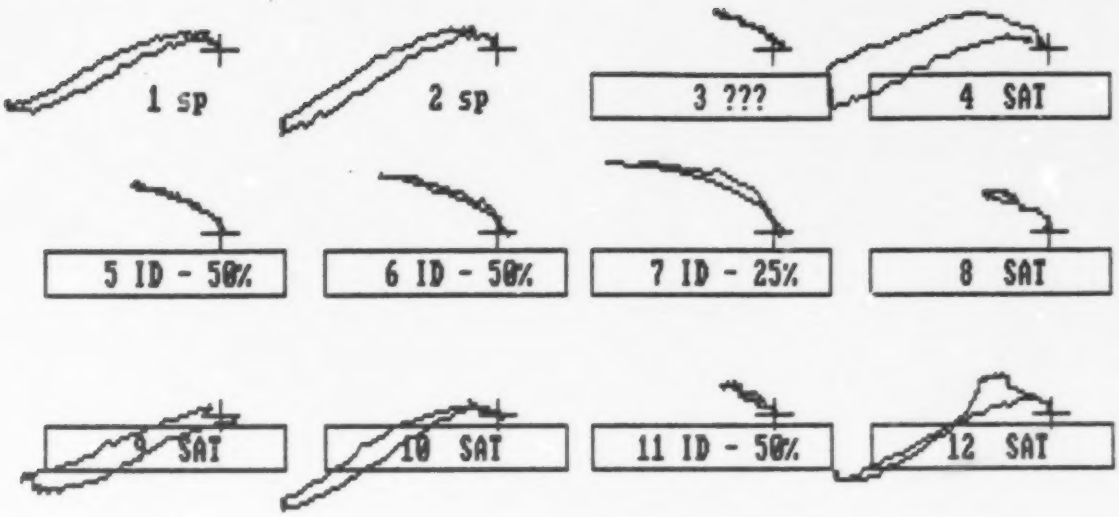
T-13-39 2 of 3 8:48:20 AM



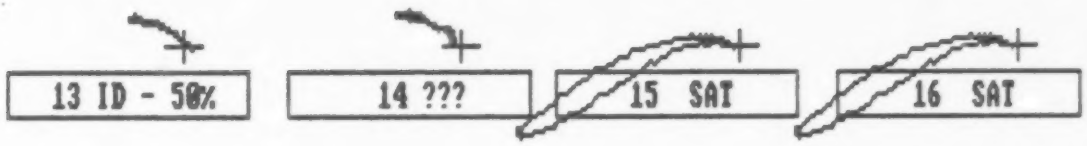

25 SAT
26 ???
27 ???
28 SAT

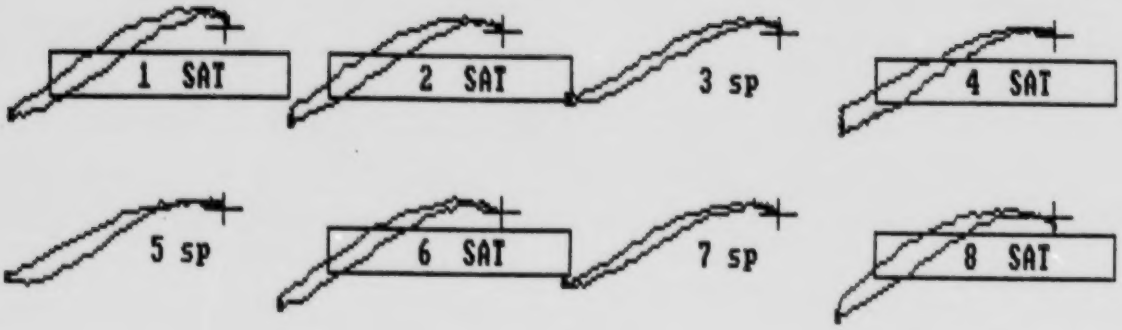

29 SAT

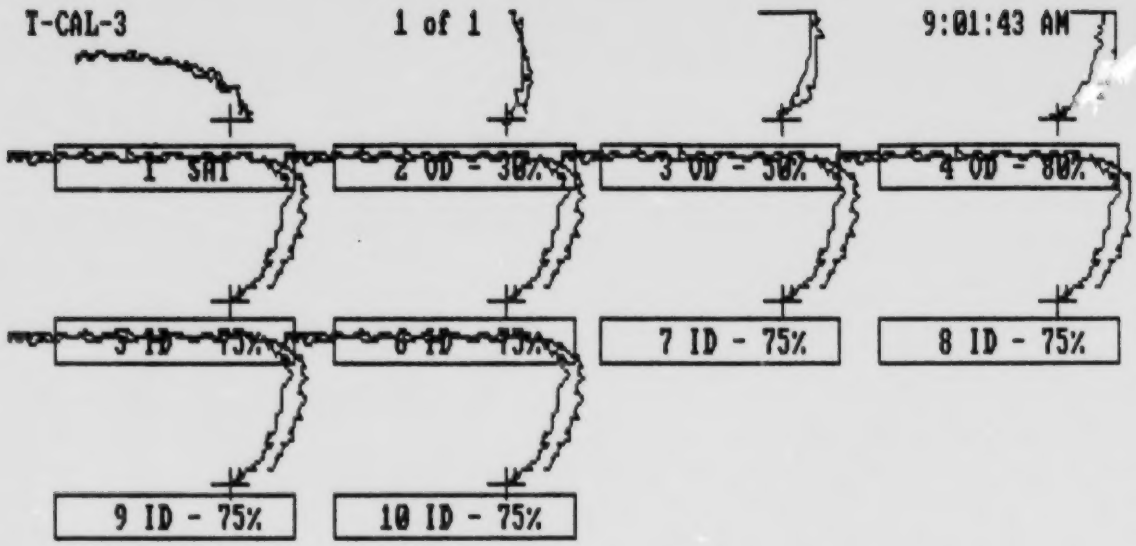
T-25-25 1 of 2 8:53:48 AM



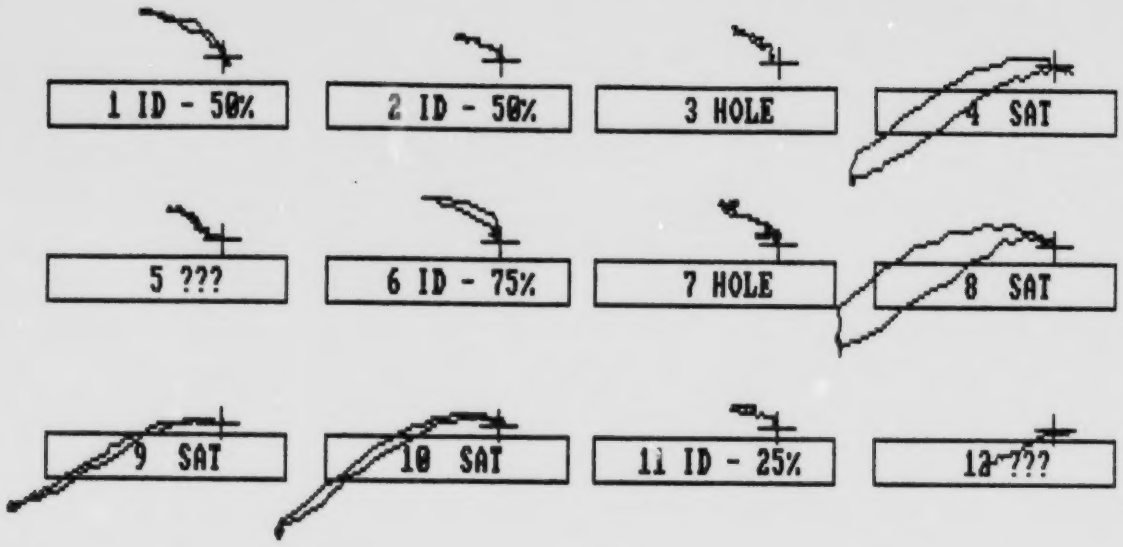
T-25-25 2 of 2 8:56:36 AM



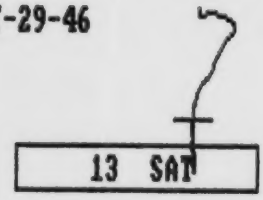

1 SAT
2 SAT
3 sp
4 SAT
5 sp
6 SAT
7 sp
8 SAT


T-CAL-3
1 of 1
9:01:43 AM
1 SAI
2 OD - 30%
3 OD - 50%
4 OD - 80%
5 ID - 75%
6 ID - 75%
7 ID - 75%
8 ID - 75%
9 ID - 75%
10 ID - 75%

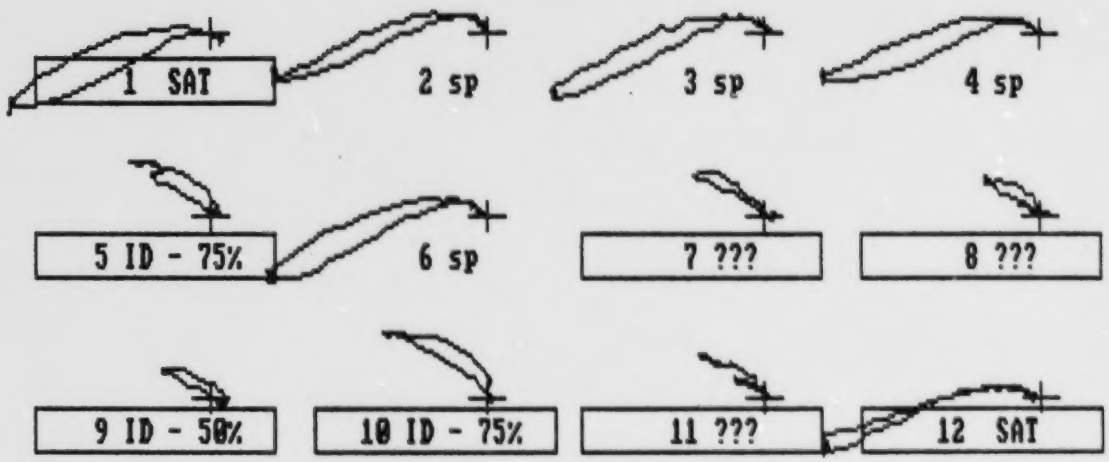
T-29-46 1 of 2 9:04:59 AM



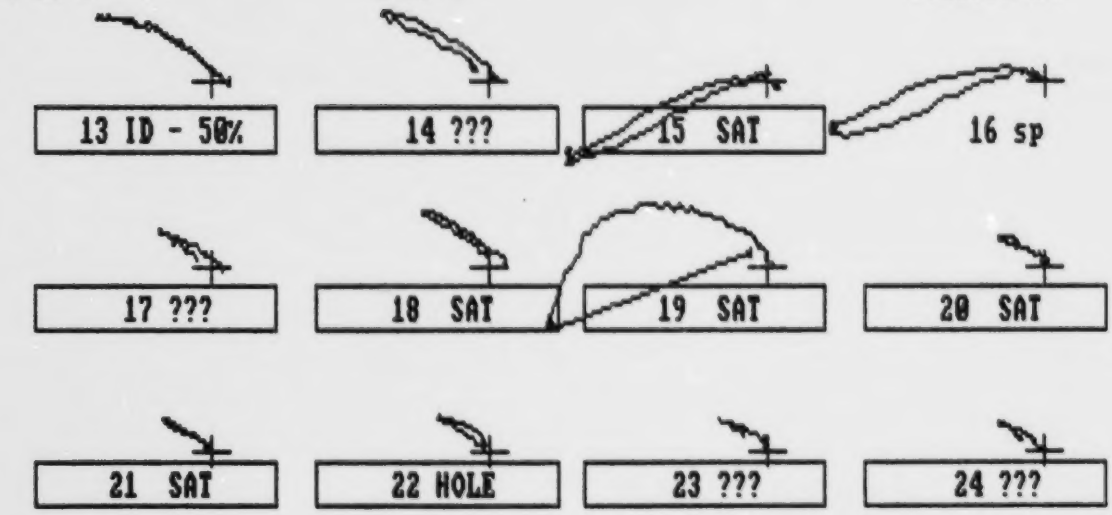
T-29-46 2 of 2 9:07:49 AM

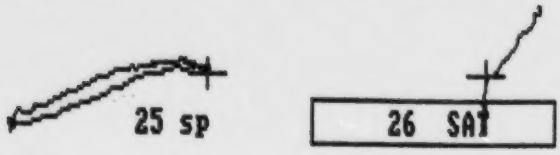


T-33-5 1 of 3 9:09:59 AM

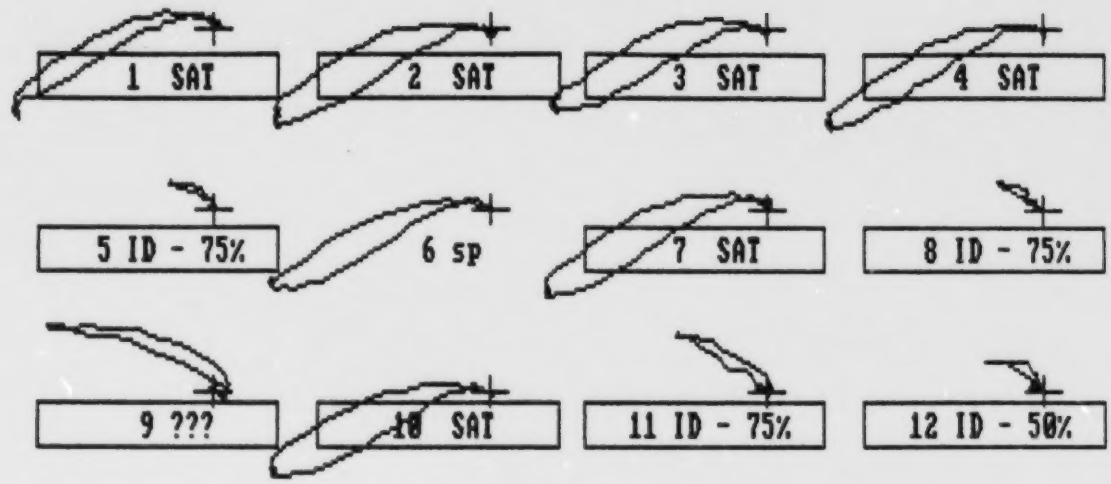


T-33-5 2 of 3 9:12:38 AM

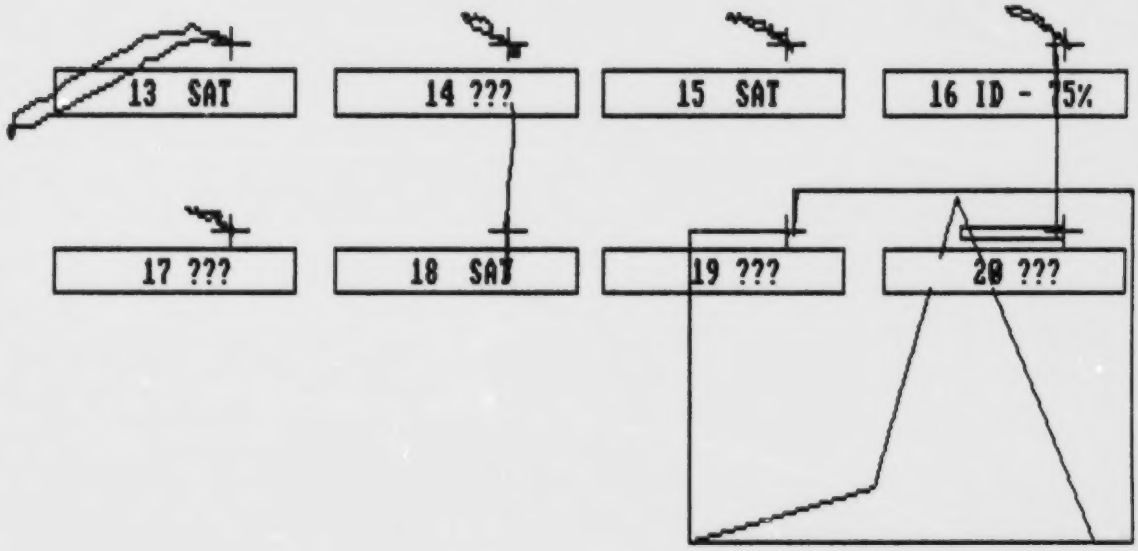


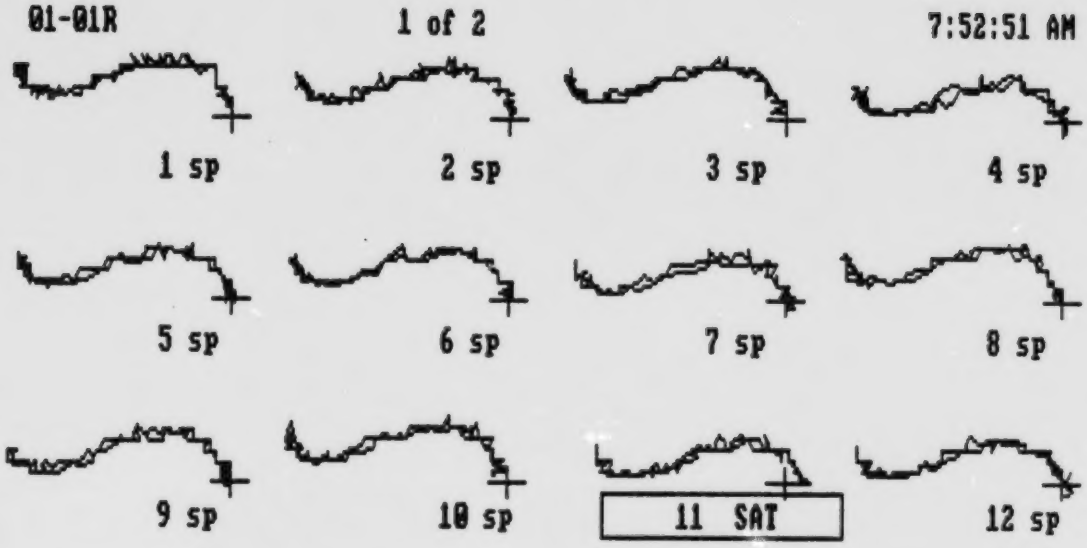

25 sp
26 SAT

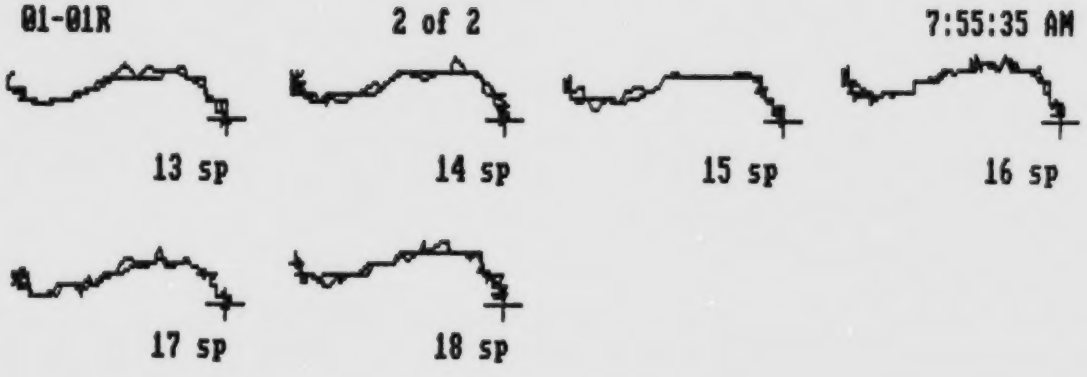
T-33-26 1 of 2 9:17:26 AM



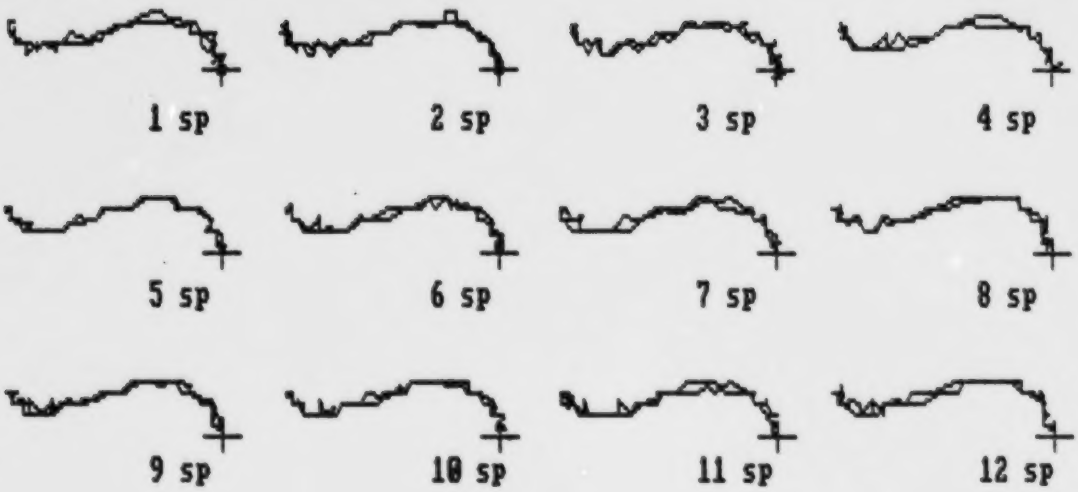
T-33-26 2 of 2 9:20:11 AM



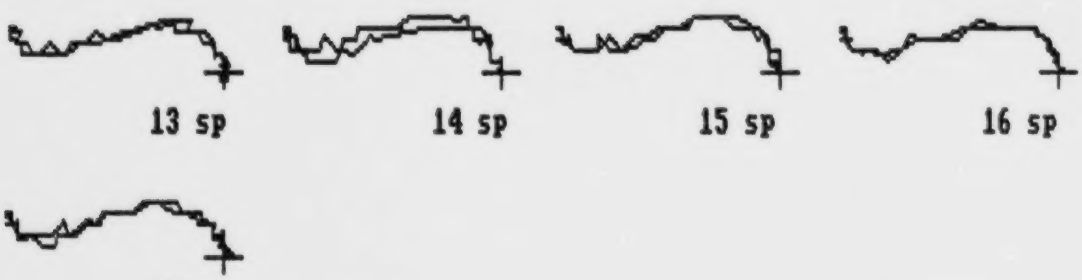

01-01R
1 of 2
7:52:51 AM
1 sp
2 sp
3 sp
4 sp
5 sp
6 sp
7 sp
8 sp
9 sp
10 sp
11 SAT
12 sp


01-01R
2 of 2
7:55:35 AM
13 sp
14 sp
15 sp
16 sp
17 sp
18 sp

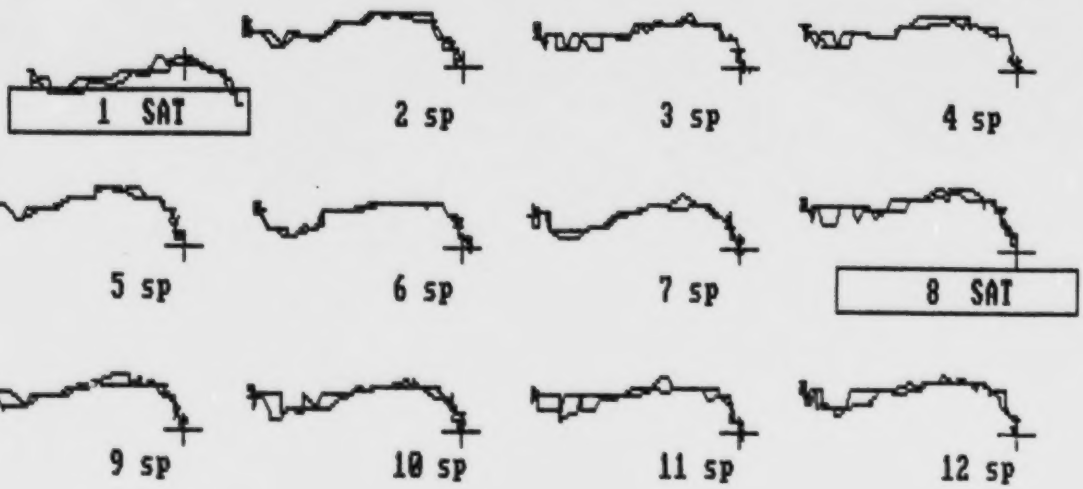
01-05 1 of 2 7:58:14 AM



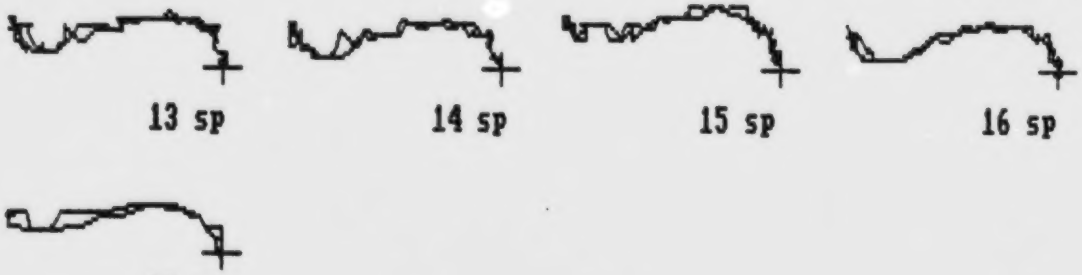
01-05 2 of 2 8:00:49 AM

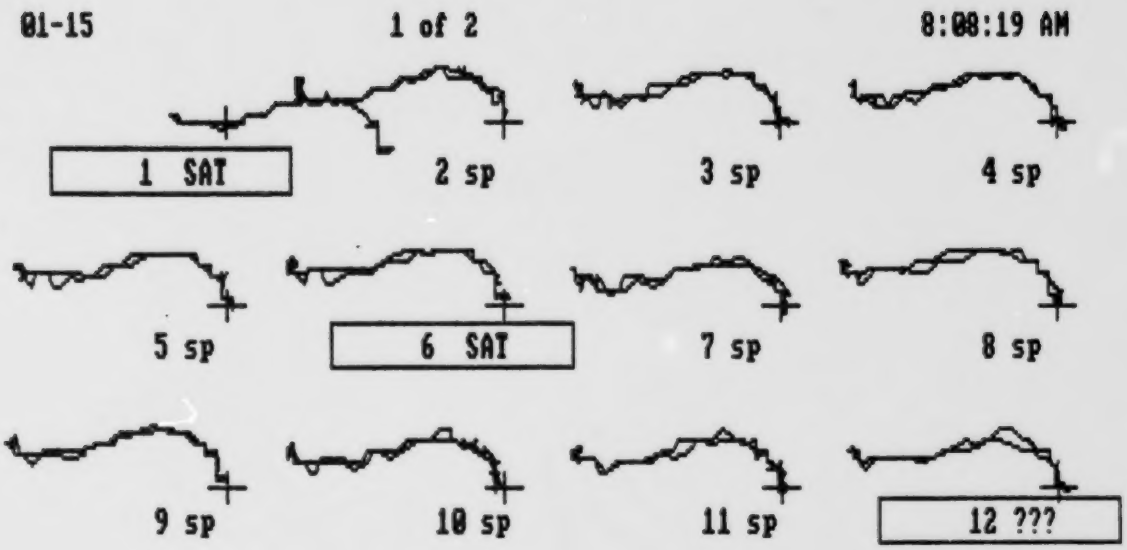


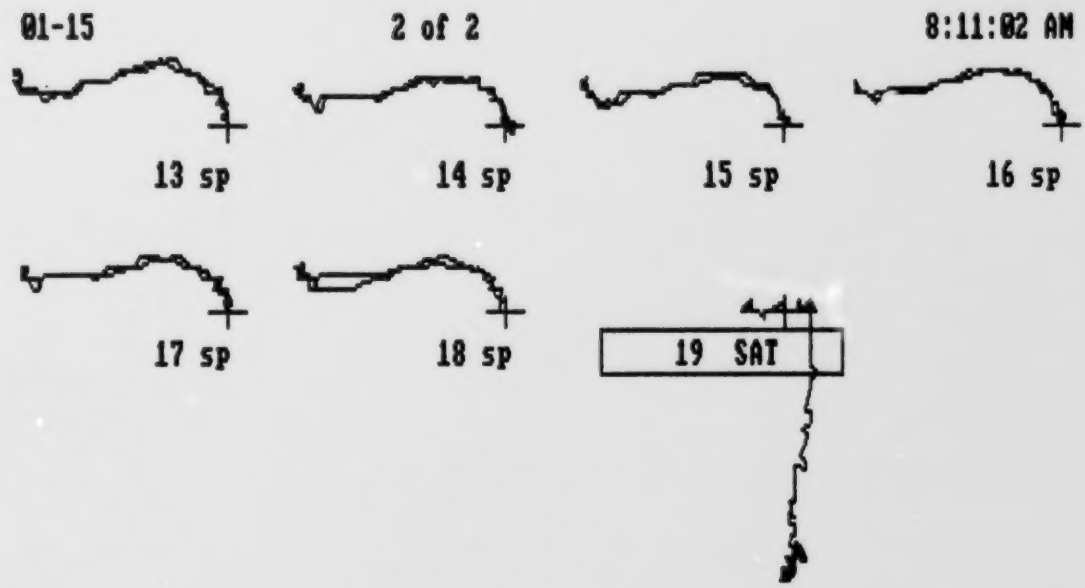
01-10 1 of 2 8:03:13 AM

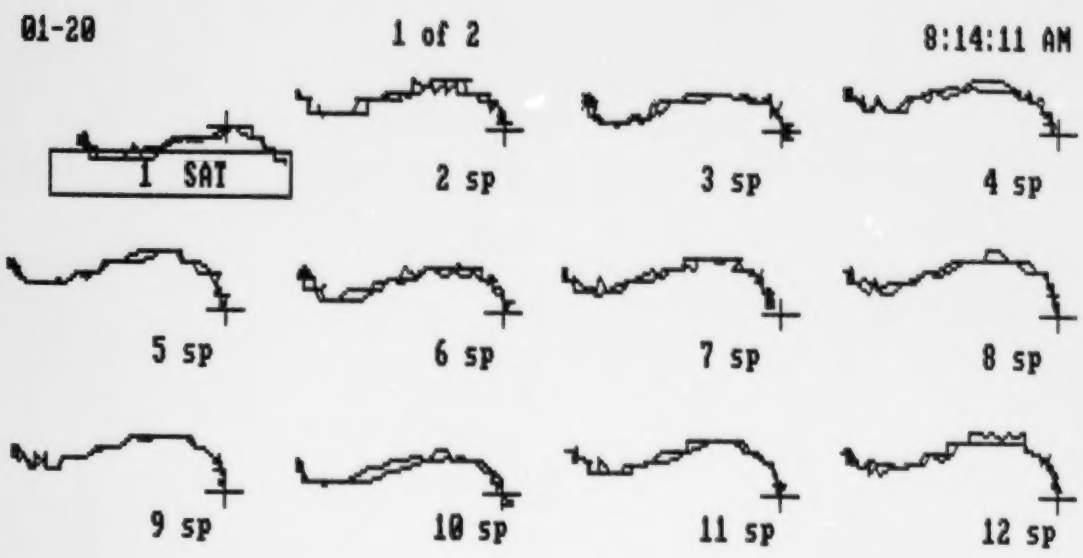


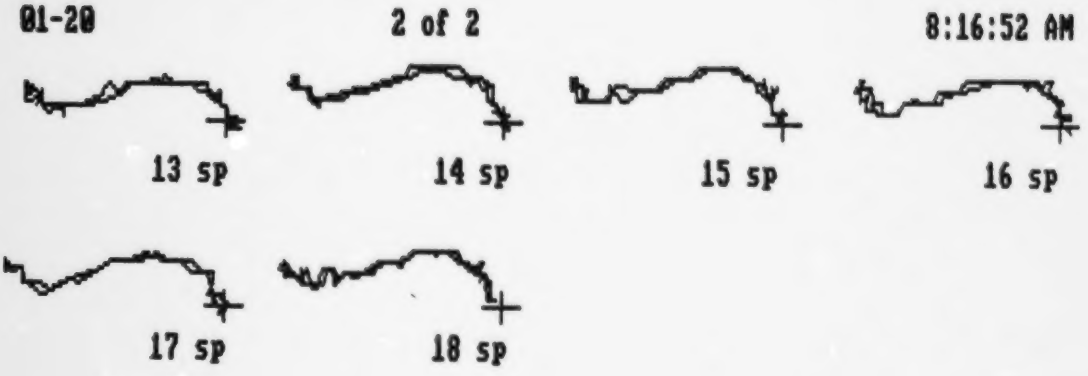
01-10 2 of 2 8:05:52 AM

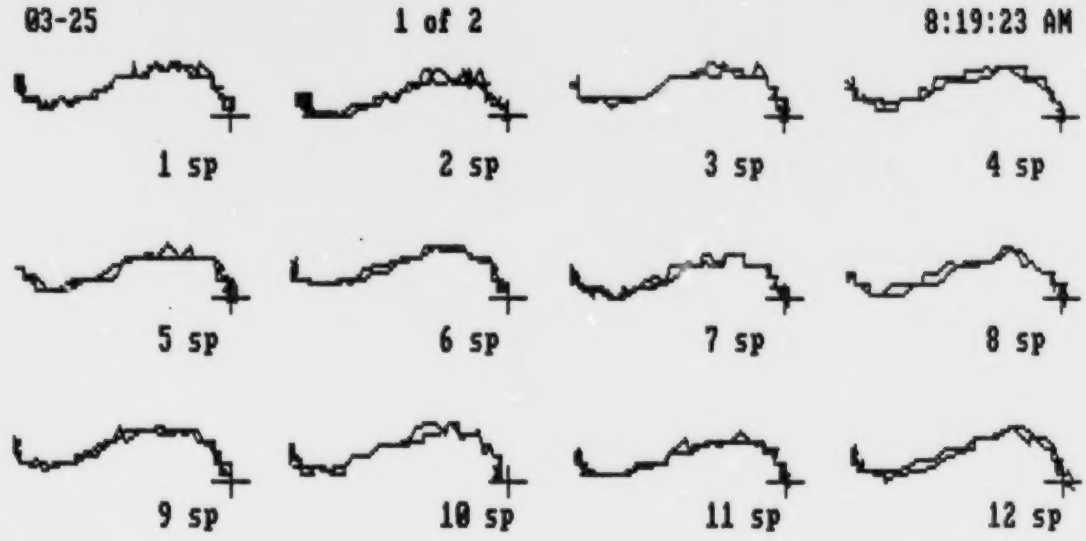


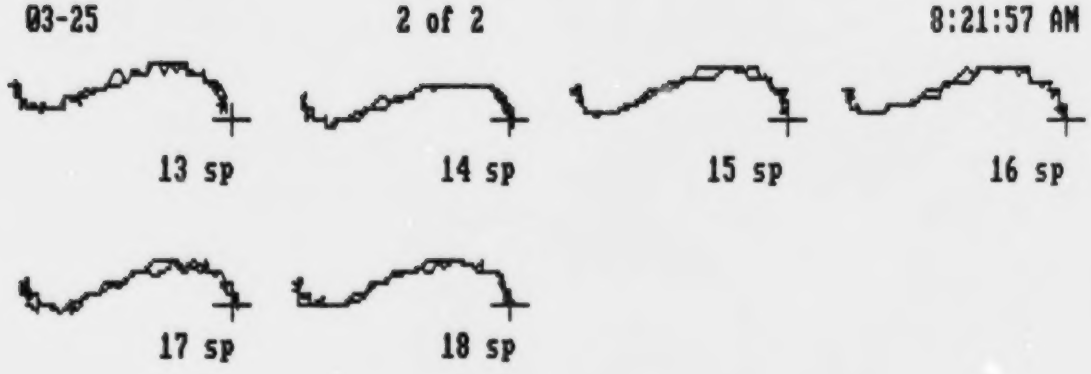

01-15
1 of 2
8:08:19 AM
1 SAT
2 sp
3 sp
4 sp
5 sp
6 SAT
7 sp
8 sp
9 sp
10 sp
11 sp
12 ???

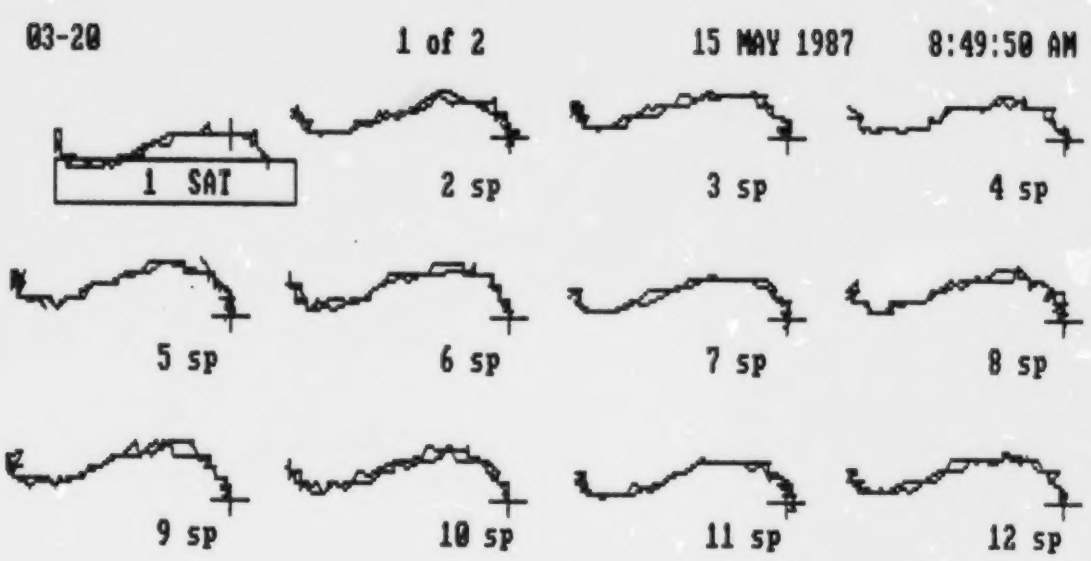

01-15
2 of 2
8:11:02 AM
13 sp
14 sp
15 sp
16 sp
17 sp
18 sp
19 SAT

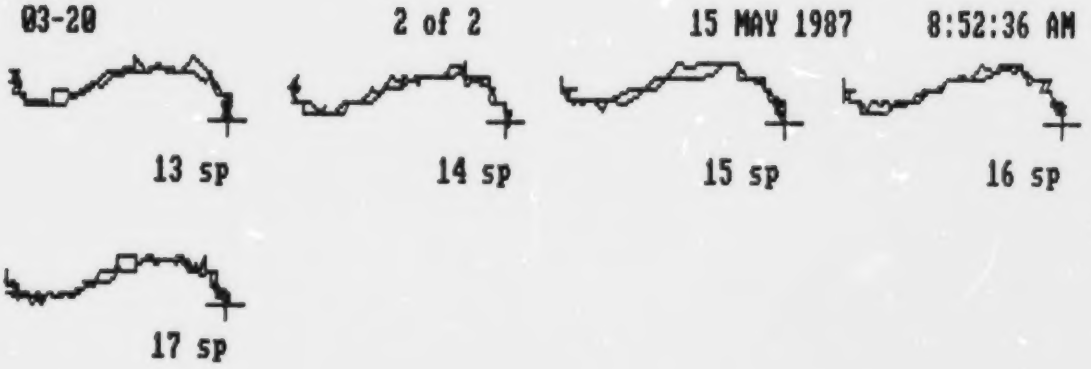

01-20
1 of 2
8:14:11 AM
1 SAT
2 sp
3 sp
4 sp
5 sp
6 sp
7 sp
8 sp
9 sp
10 sp
11 sp
12 sp


01-20
2 of 2
8:16:52 AM
13 sp
14 sp
15 sp
16 sp
17 sp
18 sp


03-25
1 of 2
8:19:23 AM
1 sp
2 sp
3 sp
4 sp
5 sp
6 sp
7 sp
8 sp
9 sp
10 sp
11 sp
12 sp


03-25
2 of 2
8:21:57 AM
13 sp
14 sp
15 sp
16 sp
17 sp
18 sp


03-20
1 of 2
15 MAY 1987
8:49:50 AM
1 SAT
2 sp
3 sp
4 sp
5 sp
6 sp
7 sp
8 sp
9 sp
10 sp
11 sp
12 sp


03-20
2 of 2
15 MAY 1987
8:52:36 AM
13 sp
14 sp
15 sp
16 sp
17 sp

APPENDIX D

CNS ANALYSIS OF THE TUBES T-... OF TABLE 4.1

EVAL FOR A.I. Project AL. XECI (12)

PNGS B - U#5 R.C.W. HX 502 JAN. /86

TAPE #2. TUBING: 90/10 Cu/Ni
5/8" OD by 0.049" wall

TAPE SPEED - 2.4 CM/SEC
X - CHANNEL - 2
Y - CHANNEL - 1
TEST FREQ. - 25 KHZ. ABSOLUTE PROBE

TAPE COUNTER	TUBE NO.	ID - INDICATION
401	25-25	MEDIUM AMP. 50% LARGE AMP. 50%
448	35-30	SMALL AMP. 50%
616	CALIBRATION	
1061	29-46	LARGE AMP. 75%
1121	33-5	MULTIPLE LARGE AMP., 75%
1269	33-26	LARGE AMP. 50%

TAPE #1
TAPE SPEED - 2.4 CM/SEC
X - CHANNEL - 2
Y - CHANNEL - 1
TEST FREQ. - 25 KHZ. ABSOLUTE PROBE

TAPE COUNTER	TUBE NO.	ID INDICATION
017	CALIBRATION	
937	13-39	LARGE AMP 75% AT SUPPORT PLATE

PNGS - U#6 MOD HX-2 AT BNFD.

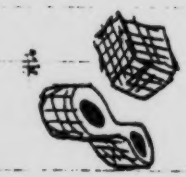
TAPE 1

TUBING – INCOLOY 800 Brno
½" OD by 0.049" wall

TAPE SPEED - 4.8 CM/SEC.
X - CHANNEL - 1
Y - CHANNEL - 2
TEST FREQUENCY - 320 KHZ.

ABSOLUTE PROBE

TAPE COUNTER	TUBE NO.	INDICATION
0400	CALIBRATION	
0959	1-11	SMALL AMPLITUDE 35% to 46% OD.
0961	1-11	POSSIBLE ID WALL LOSS.
1104	1-22	LARGE AMPLITUDE OD AT T/S.
~~1100~~	~~1-1~~	~~LARGE AMPLITUDE~~ 3
1439	2-6	LARGE AMPLITUDE OD + DENT



Volume

2

91602
new 82-10

LEGAL NOTICE

This report was prepared by Ontario Hydro and, while effort has been made to assure the accuracy and applicability of the material contained herein, neither Ontario Hydro nor any person acting on its behalf:

(a) makes any warranty, express or implied with respect to the use of any information, apparatus, method or process disclosed in this report or that such use may not infringe privately owned rights; or

(b) assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method or process disclosed in this report.

ontario hydro
research division

COMPLIMENTARY COPY
Please Credit Ontario Hydro Research if Reproduced in Whole or in Part

EXPECT - EXPERT SYSTEM FOR EDDY CURRENT TESTING

Research Report No 87-173-K
PART II

S.B. Nickerson
Theoretical Physicist
Science Section
Electrical Research Department

This consists of program listings and is available upon request.

job	file	date	report no.
16-203	819.322	August 12, 1988	87-173-K

EXPECT - EXPERT SYSTEM FOR EDDY CURRENT TESTING

S.B. Nickerson

PART II

This consists of program listings and is available upon request.

APPENDIX E

UTILITY FILES

```
; LIST OF UTILITY, PROCEDURE, RULE AND TUBE NAME FILES FOR EXPECT

; UTILITY-LIST
; LIST OF EXPECT RELATED UTILITIES
; THE FILE CONTAINING THE UTILITY HAS THE SAME NAME WITH A .LSP (LISP)
;  OR .FAS (COMPILED LISP) EXTENSION
; LOADER LOADS ALL THE EXPECT .LSP (LISP) FILES NEEDED TO FILL A
;   TUBE STRUCTURE AND PLOT THE RESULTS
; LOAFER LOADS ALL THE EXPECT .FAS (COMPILED LISP) FILES NEEDED TO FILL A
;   TUBE STRUCTURE AND PLOT THE RESULTS
; DO-ALL CONTAINS VARIOUS UTILITIES TO PRINT OR COMPILE ETC. A WHOLE
;  LIST OF PROCEDURES AT THE SAME TIME
; PLOTTUBE WILL PLOT A TUBE DATA FILE ON THE HP THINKJET
; READ-22 WILL READ TUBE DATA FILES OF UNKNOWN LENGTH AND ALL IN ONE
;  BIG DATA FILE AND SEPARATE THEM, PUTTING EACH INTO ITS OWN DATA FILE.
;  THE NAMES OF EACH FILE MUST BE KNOWN.
; ADDER CONTAINTS THE UTILITIES USED TO PRODUCE T111.S ETC.
(setf utility-list '(exptlist loader loafer do-all plottube read-22 adder))

; FILE-LIST
; LIST OF PROCEDURE FILES FOR EXPECT
; THE FILE CONTAINING THE PROCEDURES HAS THE SAME NAME WITH A .LSP (LISP)
;  OR .FAS (COMPILED LISP) EXTENSION
; NOTE THAT STRUCT SHOULD BE FOLLOWED BY TUBE-GET AND BOTH SHOULD BE BEFORE
;  ANY PROCEDURE THAT REFERENCES THE STRUCTURE IN STRUCT.  OTHERWISE THE
;  COMPILER WILL GIVE ERRORS - SEE GCLISPLM 2.2 COMPILER DOCUMENTATION
(setf file-list '(haloinit analyze struct tube-get paramtrs info
                  event limits calcprop nosp non-nosp analysis
                  display rule))

; RULE-LIST
; LIST OF THE RULE STRUCTURES
; THE FILE CONTAINING THE RULE HAS THE SAME NAME WITH A .LSP OR .FAS EXTENSION
(setf rule-list '(prosp-id nosp-id od-id id-id hole-id dent-id
                  whal-id wlspc-id scall-id sat-id hafsp-id
                  sfret-id))

; TUBE-LIST
; LIST OF ALL THE TUBES
; NOTE THAT T-1-22 IS LEFT OUT BECAUSE IT CONTAINS GARBAGE
; T111 IS A MODIFIED VERSION OF T-1-11: IT CONTAINS THE SUPERPOSITION OF EVENT
;  2 ON EVENT 1
(setf tube-list '(a999 01-01r 03-20 01-05 01-10 01-15 01-20 03-25 t-cal-0
                  t-1-11 t-1-22r t-2-6 t-cal-1 t-cal-2 t-13-39 t-25-25
                  t-25-30 t-cal-3 t-29-46 t-33-5 t-33-26 t-cal-4 t-cal-5
                  t-51-1 t111))
```

```
; THIS FILE, WHEN LOADED, LOADS ALL THE UTILITY FILES, PROCEDURE FILES
;  AND RULE FILES NECESSARY TO RUN EXPECT

; LOAD UTILITY-LIST, RULE-LIST AND FILE-LIST
; LD IS NOT A GCLISP PRIMATIVE BUT A ROUTINE THAT LOOKS IN THE DEFAULT DRIVE
;  FOR EXPTLIST.LSP AND THEN "LOADS" IT.  IF IT IS NOT THERE THEN IT
;  LOOKS THROUGH THE REST OF THE SUBDIRECTORIES UNDER D:\LSP-2
;  AND LOADS THE FIRST FILE IT FINDS CALLED EXPTLIST.LSP
(ld 'exptlist)

; LD-FILES
; LOADS THE .LSP FILES IN LIST
(defun ld-files (list)
  (dotimes (i (length list))
    (ld (nth i list))))

; LOAD THE FILES IN UTILITY-LIST - EXCEPT LOADER AND LOAFER
(ld-files (cdddr utility-list))

; LOAD THE FILES IN FILE-LIST
(ld-files file-list)

; LOAD THE FILES IN RULE-LIST
(ld-files rule-list)
```

```
; THIS FILE, WHEN LOADED, LOADS ALL THE COMPILED UTILITY FILES, COMPILED
;  PROCEDURE FILES AND COMPILED RULE FILES NECESSARY TO RUN EXPECT

; LOAD UTILITY-LIST, RULE-LIST AND FILE-LIST
; LD IS NOT A GCLISP PRIMATIVE BUT A ROUTINE THAT LOOKS IN THE DEFAULT DRIVE
;  FOR EXPTLIST.LSP AND THEN "LOADS" IT.  IF IT IS NOT THERE THEN IT
;  LOOKS THROUGH THE REST OF THE SUBDIRECTORIES UNDER D:\LSP-2
;  AND LOADS THE FIRST FILE IT FINDS CALLED EXPTLIST.LSP
(ld 'exptlist)

; LF-FILES
; LOADS THE .FAS FILES IN LIST
(defun lf-files (list)
  (dotimes (i (length list))
    (lf (nth i list))))

; FAS LOAD THE UTILITY FILES - EXCEPT LOADER AND LOAFER
(lf-files (cdddr utility-list))

; FAS LOAD THE FILES IN FILE-LIST - EXCEPT FOR HALOINIT WHICH IS LD'D
(ld 'haloinit)
(lf-files (cdr file-list))

; FAS LOAD THE FILES IN RULE-LIST
(lf-files rule-list)
```

```
; THIS FILE CONTAINS UTILITIES FOR PRINTING OR COMPILING ALL THE EXPECT FILES
;  AND FOR ANALYZING ALL THE TUBES AT ONCE
; IT IS NECESSARY THAT LOADER BE ALREADY LOADED

; PRINT-ALL
; PRINTS ALL THE EXPECT PROCEDURE AND RULE FILES AND AN ENGLISH VERSION OF
;  ALL THE RULES TO AN HP THINKJET
(defun print-all ()
  (print-list-file utility-list)
  (print-list-file file-list)
  (print-list-file rule-list)
  (rule-writer)
  (print-list-file '(rule-eng)))

; PRINT-LIST-FILE
; PRINTS ALL THE FILES IN LIST ON THE HP LASERJET
(defun print-list-file (list)
    (setf page-new 1)
    (dotimes (i (length list))
      (setf page-new (prn-file (nth i list) page-new))))

; PRINT-ALL-LAS
; PRINTS ALL THE EXPECT PROCEDURE AND RULE FILES AND AN ENGLISH VERSION
;  OF ALL THE RULES TO AN HP LASERJET
(defun print-all-las ()
  (print-list-file-las utility-list "E")
  (print-list-file-las file-list "F")
  (print-list-file-las rule-list "G")
  (rule-writer)
  (print-list-file-las '(rule-eng) "H"))

; PRINT-LIST-FILE-LAS
; PRINTS ALL THE FILES IN LIST ON THE HP LASERJET
(defun print-list-file-las (list prefix)
    (setf page-new 1)
    (dotimes (i (length list))
      (setf page-new (prn-file-las (nth i list) page-new 'lsp prefix))))


; COMP-ALL
; COMPILES ALL THE EXPECT PROCEDURE AND RULE FILES EXCEPT HALOINIT, LOADER
;  AND LOAFER
(defun comp-all ()
  (let ((list (append (cddr utility-list) (cdr file-list) rule-list)))
    (dotimes (i (length list))
      (comp (nth i list)))))
```

```
; F-AND-C-ALL
; FILLS ALL THE SLOTS OF THE TUBES IN TUBE-LIST - IF NOT ALREADY EXISTING
(defun f-and-c-all ()
  (dotimes (i (length tube-list))
    (ignore-errors (front-and-center (nth i tube-list)))
    (erase-from-ram (nth i tube-list))))

; PLOT-ALL
; PLOTS ON THE PRINTER ALL THE EVENTS OF THE TUBES IN TUBE-LIST
(defun plot-all ()
  (dotimes (i (length tube-list))
    (ignore-errors (analyze-tube (nth i tube-list) t))))
```

```
; PRINT-DATA-FILE
; PLOTS THE FILE NAME ON AN HP THINK-JET PRINTER - SO DIP SWITCH 5 MUST
;  BE DOWN -  RETURN TO UP WHEN FINISHED TO RETURN TO EPSON MODE
(defun print-data-file
   (name &optional (points-per-point 50))       ; NO OF POINTS PER LINE
  (setf twos '(1 2 4 8 16 32 64 128))
  (data)
  (setf stream-name (open (pathname (symbol-name name)) :direction ':input
                          :element-type 'unsigned-byte))
  (setf stream-printer (open (pathname "lpt1") :direction ':output))
  (princ "                                " stream-printer)
  (princ (symbol-name name) stream-printer)
  (terpri stream-printer)
  (terpri stream-printer)
  (terpri stream-printer)
  (princ "                    " stream-printer)
  (princ "Y" stream-printer)
  (princ "                                      " stream-printer)
  (princ "X" stream-printer)
  (terpri stream-printer)
  (terpri stream-printer)
  (princ (string-append (setf esc (string 27))
                        "*r640S")               ; SEE HP MANUAL PG 2-2 FF
                        stream-printer)
  (princ (string-append esc "*rB")
         stream-printer)                        ; "
  (princ (string-append esc "*rA")
         stream-printer)                        ; "
  (read-dat 92 stream-name)                     ; TUBE ID DATA
  (catch 'up
    (loop (setf arr (make-array 256             ; STRING OF 256 "0"s
                                :element-type 'string-char
                                :initial-element 48)
                brr (make-array 256 :element-type 'string-char
                                    :initial-element 48))
          (setf xs (convert-dat (car            ; DECIMAL FROM 2's COMPLEMENT
                                                ;   RANGE 0 TO 255
                                  (setf numbs
                                    (read-dat points-per-point
                                              stream-name))))
                ys (convert-dat (cadr numbs)))
          (when (not numbs) (throw 'up nil))
          (dotimes (i points-per-point)
                   (transform arr (nth i xs)) ; CHANGE 0 TO 1 AT POSITION
                                              ;   (NTH I XS)
                   (transform brr (nth i ys)))
          (setf crr (append (shrink brr) (shrink arr)))
```

```
            (princ (string-append esc "*b64W")  ; 64 CHARACTERS COMING
                   stream-printer)
            (dotimes (i 64)
                (princ (string (nth i crr)) stream-printer))))
  (princ (string-append esc "*rB") stream-printer)
  (ignore-errors (close-all-files)))


; READ-DAT
; READS IN N DATA POINTS FRAM STREAM
; RETURNS ((X VALUES) (Y VALUES))
(defun read-dat (n stream)
  (catch 'up
    (let* ((x (make-list n :initial-element 0))
           (y (make-list n :initial-element 0)))
      (dotimes (i n)
               (setf x-value (read-char stream nil 2048))
               (when (= x-value 2048) (throw 'up nil)) ;end of file
               (setf y-value (read-char stream nil 2048))
               (when (= y-value 2048) (throw 'up nil))
               (setf (nth i x) x-value (nth i y) y-value))
      (list x y))))

; SHRINK
; TAKES A 256 LONG ARRAY OF 1s AND 0s AND CONVERTS THE FIRST 8
; ELEMENTS TO A NUMBER BETWEEN 0 AND 255 ETC.
; RETURNS A 32 LONG LIST OF NUMBERS.
; NOTE THE NOTE ON 26 BELOW
(defun shrink (array)
  (let ((list nil)
        (stream (make-string-input-stream array)))
    (dotimes (i 32)
       (setf list (cons
                        (+ (* 128 (read-char-minus stream))
                           (* 64 (read-char-minus stream))
                           (* 32 (read-char-minus stream))
                           (* 16 (read-char-minus stream))
                           (* 8 (read-char-minus stream))
                           (* 4 (read-char-minus stream))
                           (* 2 (read-char-minus stream))
                           (* 1 (read-char-minus stream))) list)))
    (close stream)
    (reverse (subst 30 26 list))))       ; SENDING A 26 TO THE PRINTER CAUSES
                                         ; A WRITE FILE ERROR

; CONVERT-DAT
; CONVERTS A LIST OF NUMBERS FROM 2s COMPLEMENT TO THE RANGE 0 TO 255
(defun convert-dat (x)
```

```
  (mapcar #'(lambda (y) (cond ((> y 127) (- y 128))
                              (T (+ y 128))))
          x))
; READ-CHAR-MINUS
; 48 is ASCII CODE FOR 0
(defun read-char-minus (stream)
 (- (read-char stream) 48))

; TRANSFORM
; MAKES NUMBERth ELEMENT OF ARRAY THE ASCII CODE FOR 1
(defun transform (array number)
  (setf (aref array number) 49))
```

```
; READS THE 22 TEST TUBES AND PUTS THEM IN SEPARATE DOS FILES
; HOWEVER SOMEWHERE ON 50-1  THERE WAS A DOS ERROR SO STOPPED THERE
; ALTHOUGH THE STREAM STAYED OPEN AND READ MORE WITH (RE N) SO YOU CAN
; RECOVER FROM THIS.

(defun read-and()
  (setf tube-lables                                ; DOS FILE NAMES
        '(t-cal-0 t-1-11 t-1-22 t-1-22r t-2-6
          t-cal-1
          t-cal-2 t-13-39 t-25-25 t-25-30 t-cal-3 t-29-46 t-33-5 t-33-26
          t-cal-4 t-cal-5 t-51-1 t-50-1 t-46-7 t-46-7R1 t-46-7R2
          t-46-7R3))
  (setf first-char                                 ; FIRST CHAR AFTER SPACE ON
                                                   ; HEADER. WRITE-NEXT-FILE
                                                   ; LOOKS FOR THIS
                '("1" "1" "1" "2"
                  "C"
                  "C" "1" "2" "2" "C" "2" "3" "3"
                  "C" "C" "5" "5" "4" "4" "4"
                  "4" "z")) ; will go to end then signal error

  (data)                                           ; SET DOS DIRECTORY
  (setf stream-a (input-stream "a:\\andesec1.dat"))
  (setf temp nil)                                  ; WRITE-NEXT-FILE DUMPS CHARS
                                                   ; HERE WHEN KEYS ON FIRST-CHAR
  (dotimes (i 22)
           (write-next-file stream-a
                            (output-stream (nth i tube-lables))
                                                   ; CHAR CHANGES STRING TO NUMBER
                            (char (nth i first-char) 0)))
  (close stream-a)
  (sys::8087-fpp :use))


(defun write-next-file (in-stream out-stream ch)
                                                   ; FIRST 12 CHARS FROM HEADER
  (when (> (length temp) 0) (dotimes (i (length temp))
                                     (write-char (nth i temp) out-stream)))
  (catch 'up (loop (setf ch1 (read-char in-stream) ch2 (read-char in-stream))
                                                   ;KEY ON SPACE
                   (cond ((/= ch1 32) (write-char ch1 out-stream)
                                      (write-char ch2 out-stream))
                                                   ; KEY ON FIRST-CHAR
                         ((/= ch2 ch) (write-char ch1 out-stream)
                                      (write-char ch2 out-stream))
                                                   ; IF CH1 CH2 IS FOUND, GRAB NEXT
                                                   ; 10 CHARS, PRINT TO THE CONSOL
                                                   ; AND ASK IF A NEW HEADER HAS
```

```
                                        ; BEEN FOUND: Y => IT HAS
                                        ;             N => IT HAS NOT
                          (T (setf temp (append (list ch1 ch2)
                                        ; READ 10 CHARS
                                        (read-bytes in-stream 10)))
                             (print (letters temp) *terminal-io*)
                                  ; Y => EXIT AND START NEW FILE
                             (cond ((y-or-n-p) (throw 'up nil))
                                  ; N => WRITE THE 12 CHARS AND CONTINUE ON
                                   (T (dotimes (i 12)
                                        (write-char (nth i temp)
                                                    out-stream))))))))
  (close out-stream)
  (sys::8087-fpp :use))


; LETTERS
;  A LIST OF ASCII NUMBERS IS RETURNED AS A STRING OF CHARS
(defun letters (list)
  (setf lett "")
  (dotimes (i (length list))
           (setf lett (string-append lett (string (nth i list)))))
  lett)


; READ-BYTES
; READS N BYTES FROM IN-STREAM AND RETURNS THEM AS A LIST OF ASCII NUMBERS
(defun read-bytes (in-stream n)
  (setf list nil)
  (dotimes (i n)
           (setf list (cons (read-char in-stream) list)))
  (reverse list))



; CONVERT
; WITH THW ANDES HEADER IN LIST THE FILE LENGTH IS RETURNED - NOT USED HERE
(defun convert (list)
  (+ (nth 180 list)
     (* 256 (+ (nth 181 list)
               (* 256 (+ (nth 182 list)
                         (* 256 (nth 183 list))))))))


; INPUT-STREAM
; OPENS AN INPUT FILE
; NAME MUST BE A STRING
(defun input-stream (name)
```

```
  (open (pathname name) :direction ':input
                                   :element-type 'unsigned-byte))


; OUTPUT-STREAM
; OPENS AN OUTPUT-STREAM IN THE DEFAULT DIRECTORY
; NAME MUST BE A SYMBOL
(defun output-stream (name)
  (open (pathname (symbol-name name)) :direction ':output
                                      :element-type 'unsigned-byte))


; RE
; READ N BYTES FROM STREAM-A
(defun re (n)
  (read-bytes stream-a n))
```

```
; ADDER
; ADDS TWO LISTS TOGETHER WITH ELEMENT 0 OF LIST2 BEING ADDED TO ELEMENT
;  PLACE OF LIST1
; IF PLACE IS LESS THAN 0 THEN 0'S ARE ADDED TO THE FRONT OF LIST1 TO
;  ACCOMADATE AND SIMILARLY AT THE END OF LIST1
(defun adder (list1 list2 place)
  (setf len1 (length list1)
        len2 (length list2)
        list-back (copy-list list1))
  (when (< place 0)
        (setf list-back (append (make-list (abs place) :initial-element 0)
                                list-back))
        (setf place 0))
  (when (> (+ len2 place) len1)
        (setf list-back (append list-back
                                (make-list (- (+ len2 place) len1)
                                           :initial-element 0))))
  (dotimes (i len2)
    (setf (nth (+ place i) list-back)
          (+ (nth i list2) (nth (+ place i) list-back))))
  list-back)


; T111-MAKE
; USED TO MAKE FILE T111.S
; ADDS FIRST AND SECOND EVENTS OF T-1-11
; TO USE:
;       1. MAKE A COPY OF T-1-11 NAMED T111
;       2. USE WS2000 TO CHANGE THE (SETF T-1-11 ...)
;          TO (SETF T111 ...)
;       3. USE WS2000 TO CREATE AN EMPTY T111
;       4. FRONT-AND-CENTER T-1-11.S AND T111.S
;       5. (T111-MAKE)
;       6. USE PUT-VALUE TO REPLACE X AND Y WITH NEW-X AND NEW-Y IN T111.S
;       7. (SAVE-TUBE 'T111)
;       8. (FRONT-AND-CENTER 'T111 'LIMITS)
;       9. (AT 'T111) WITH BOX AND ANALYSIS DISABLED IN DISPLAY
(defun t111-make ()
  (setf a (get-value t-1-11 x))
  (setf b (get-value t-1-11 y))
  (setf x0 (nth 0 a))
  (setf y0 (nth 0 b))
  (setf x1 (nth 1 a))
  (setf y1 (nth 1 b))
  (setf new-x nil new-y nil)
  (dotimes (i 11)
    (setf new-x (cons (adder x0 x1 (- (* i 20) 60)) new-x))
    (setf new-y (cons (adder y0 y1 (- (* i 20) 60)) new-y)))
```

```
  (setf new-x (reverse new-x))
  (setf new-y (reverse new-y)))

; T112-MAKE
; USED TO MAKE FILE T112.S
; ADD EVENT 0 OF T-1-11 AND EVENT 2 OF T-29-46
; SEE COMMENTS ON USE WITH T111-MAKE
(defun t112-make ()
  (setf a (get-value t-1-11 x))
  (setf b (get-value t-1-11 y))
  (setf x0 (nth 0 a))
  (setf y0 (nth 0 b))
  (setf a (get-value t-29-46 x))
  (setf b (get-value t-29-46 y))
  (setf x1 (nth 2 a))
  (setf y1 (nth 2 b))
  (setf new-x nil new-y nil)
  (dotimes (i 11)
    (setf new-x (cons (adder x0 x1 (- (* i 20) 60)) new-x))
    (setf new-y (cons (adder y0 y1 (- (* i 20) 60)) new-y)))
  (setf new-x (reverse new-x))
  (setf new-y (reverse new-y)))

; T113-MAKE
; USED TO MAKE FILE T113.S
; ADD EVENT 0 OF T-1-11 AND (EVENT 2 OF T-29-46 WITH ALL VALUES MULTIPLIED
;  BY 2)
; SEE COMMENTS ON USE WITH T111-MAKE
(defun t113-make ()
  (setf a (get-value t-1-11 x))
  (setf b (get-value t-1-11 y))
  (setf x0 (nth 0 a))
  (setf y0 (nth 0 b))
  (setf a (get-value t-29-46 x))
  (setf b (get-value t-29-46 y))
  (setf x1 (mapcar #'(lambda(x) (* 2 x)) (nth 2 a)))
  (setf y1 (mapcar #'(lambda(x) (* 2 x)) (nth 2 b)))
  (setf new-x nil new-y nil)
  (dotimes (i 11)
    (setf new-x (cons (adder x0 x1 (- (* i 20) 60)) new-x))
    (setf new-y (cons (adder y0 y1 (- (* i 20) 60)) new-y)))
  (setf new-x (reverse new-x))
  (setf new-y (reverse new-y)))


; T114-MAKE
; USED TO MAKE FILE T114.S
; ADD EVENT 1 OF T-1-11 AND EVENT 2 OF T-29-46
```

```
; SEE COMMENTS ON USE WITH T111-MAKE
(defun t114-make ()
  (setf a (get-value t-1-11 x))
  (setf b (get-value t-1-11 y))
  (setf x0 (nth 1 a))
  (setf y0 (nth 1 b))
  (setf a (get-value t-29-46 x))
  (setf b (get-value t-29-46 y))
  (setf x1 (nth 2 a))
  (setf y1 (nth 2 b))
  (setf new-x nil new-y nil)
  (dotimes (i 11)
    (setf new-x (cons (adder x0 x1 (- (* i 10) 45)) new-x))
    (setf new-y (cons (adder y0 y1 (- (* i 10) 45)) new-y)))
  (setf new-x (reverse new-x))
  (setf new-y (reverse new-y))
  (front-and-center 't114)
  (put-value t114 x new-x)
  (put-value t114 y new-y)
  (save-tube 't114)
  (front-and-center 't114 'limits))
```

APPENDIX F

EXPECT FILES

```
; THIS FILE LOADS THE FILES NECESSARY TO RUN THE HALO GRAPHICS ROUTINES AND
;  SET THE PARAMETERS FOR AN IBM COLOR GRAPHICS ADAPTER AND HP THINKJET

  (cd "c:\\halo")
  (load "halofix")
  (load "lm-conv")
  (cd "d:\\lsp-2")
  (use-package 'halo)
  (setdev "c:\\halo\\haloibm.dev")
  (setprn "c:\\halo\\halotjet.prn")
  (setpattr '(0 0 0 1 1 0 1 0))                    ; SCALE FOR GRAPHICS DUMP
                                                   ;  X,Y = 1,1
                                                   ; NO AUTOMATIC FORM FEED
```

```
; AT
; NICKNAME FOR ANALYZE-TUBE
; NAME IS THE TUBE NAME
; IF SLOT IS NON-NIL THEN SLOTS STARTING WITH SLOT SLOT ARE RE-FILLED
; HOWEVER NOTE THAT SOME OF THE CALCULATED-PROPERTIES ARE CALCULATED
;   FROM NOSP-AVERAGES, FEATURES AND ANALYSIS - AS NEEDED, AND SIMILARLY
;   FOR FEATURES.
(defun at (name &optional (slot nil) (plot-results? nil))
  (analyze-tube name plot-results? slot))

; ANALYZE-TUBE
; SEE AT
(defun analyze-tube (name &optional (plot-results? nil) (slot nil))
  (front-and-center name slot)
  (display-results name plot-results?)
  (closegraphics)
  (erase-from-ram name))

; ERASE-FROM-RAM
(defun erase-from-ram (name)
  (setf (symbol-value name) nil))
```

```
; THIS FILE CONTAINS ALL THE PROCEDURES FOR DEFINING AND MANIPULATING THE
;  DATA STRUCTURE THAT CONTAINS THE TUBE.
; NOTE THAT THE TUBE STRUCTURE IS LAST SO THAT ITS DEFAULT SLOT VALUE WILL BE
;  DEFINED.
; THIS FILE ALSO CONTAINS FRONT-AND-CENTER (AND ASSOCIATED PROCEDURES).
;  FRONT-AND-CENTER FILLS THE DATA STRUCTURE FOR A GIVEN
;  TUBE - READING OFF DISK OR (RE)CALCULATING VALUES AS NECESSARY.
; THE SLOTS FOR A GIVEN TUBE ARE FILLED IN THE ORDER IN WHICH THEY ARE DEFINED.


; MACROS PUT HERE FOR PROPER COMPILATION - SEE GCLISP LM 2.2 COMPILER
;  DOCUMENTATION.

; GET-VALUE
; RETURNS ANY VALUE FROM TUBE AND RELATED STRUCTURES WITH ..-EDD MACROS
; STRUCTURE CONTAINS THE STRUCTURE (NOT ITS NAME) AND VALUE-NAME
;  CONTAINS THE KEYWORD
(defmacro get-value (structure value-name)
  `(,(read-from-string (string-append (string value-name) "-edd"))
     ,structure))


; REPLACES ANY VALUE FROM TUBE AND RELATED STRUCTURES
; STRUCTURE CONTAINS THE STRUCTURE (NOT ITS NAME) AND VALUE-NAME
;  CONTAINS THE KEYWORD
(defmacro put-value (structure value-name &optional (value nil))
   `(setf (,(read-from-string (string-append (string value-name) "-edd"))
           ,structure) ,value))


; GET-PROCEDURE
; RETURNS PROCEDURE NAME
(defmacro get-procedure (slot)
  `(get ,slot 'procedure))


;-------------------------------------------------------------------------
;                  PARAMETERS STRUCTURE AND PROCEDURES                    |
;-------------------------------------------------------------------------

; PARAMETERS
; FOR ALL STRUCTURES
; SEE PARAMTRS.LSP FOR DOCUMENTATION
(defstruct (parameters)
  (or-trigger nil)
  (and-trigger nil)
```

```
   (or-minimum nil)
   (and-minimum nil)
   (and-minimum-length nil)
   (angles nil)
   (tip nil)
   (basis-length nil)
   (length-comp nil)
   (diff-vect nil)
   (nosp-length nil)
   (nosp-angle nil)
   (non-n-basis-length nil))

; PARAMETERS-FILL IS THE NAME OF THE PROCEDURE THAT FILLS PARAMETERS.
;  IT IS CANTAINED IN PARAMTRS.LSP
(setf (get 'parameters 'procedure) 'parameters-fill)

; OR-TRIGGER-EDD
(defmacro or-trigger-edd (name)
  `(parameters-or-trigger (tube-parameters ,name)))

; AND-TRIGGER-EDD
(defmacro and-trigger-edd (name)
  `(parameters-and-trigger (tube-parameters ,name)))

; OR-MINIMUM-EDD
(defmacro or-minimum-edd (name)
  `(parameters-or-minimum (tube-parameters ,name)))

; AND-MINIMUM-EDD
(defmacro and-minimum-edd (name)
  `(parameters-and-minimum (tube-parameters ,name)))

; AND-MINIMUM-LENGTH-EDD
(defmacro and-minimum-length-edd (name)
  `(parameters-and-minimum-length (tube-parameters ,name)))

; ANGLES-EDD
(defmacro angles-edd (name)
  `(parameters-angles (tube-parameters ,name)))

; TIP-EDD
(defmacro tip-edd (name)
  `(parameters-tip (tube-parameters ,name)))

;BASIS-LENGTH-EDD
(defmacro basis-length-edd (name)
  `(parameters-basis-length (tube-parameters ,name)))
```

```
; LENGTH-COMP-EDD
(defmacro length-comp-edd (name)
  `(parameters-length-comp (tube-parameters ,name)))

; DIFF-VECT-EDD
(defmacro diff-vect-edd (name)
  `(parameters-diff-vect (tube-parameters ,name)))

; NOSP-LENGTH-EDD
(defmacro nosp-length-edd (name)
  `(parameters-nosp-length (tube-parameters ,name)))

; NOSP-ANGLE-EDD
(defmacro nosp-angle-edd (name)
  `(parameters-nosp-angle (tube-parameters ,name)))

;NON-N-BASIS-LENGTH-EDD
(defmacro non-n-basis-length-edd (name)
  `(parameters-non-n-basis-length (tube-parameters ,name)))


;-------------------------------------------------------------------------
;                    INFO STRUCTURE AND PROCEDURES                        |
;-------------------------------------------------------------------------

; INFO
; INFORMATION IDENTIFYING TUBE - COPIED OFF  THE ANDES HEADER
(defstruct (info)
  (tube-number nil)
  (station nil)
  (unit nil)
  (heat-exchanger-type nil)
  (heat-exchanger-number nil)
  (frequency nil)
  (date nil)
  (time nil)
  (supervisor nil)
  (comment nil))


; INFO-FILL IS THE NAME OF THE PROCEDURE THAT FILLS INFO.
;  IT IS CONTAINED IN INFO.LSP
(setf (get 'info 'procedure) 'info-fill)

; TUBE-NUMBER-EDD
(defmacro tube-number-edd (name)
  `(info-tube-number (tube-info ,name)))
```

```
; STATION-EDD
(defmacro station-edd (name)
  `(info-station (tube-info ,name)))

; UNIT-EDD
(defmacro unit-edd (name)
  `(info-unit (tube-info ,name)))

; HEAT-EXCHANGER-TYPE-EDD
(defmacro heat-exchanger-type-edd (name)
  `(info-heat-exchanger-type (tube-info ,name)))

; HEAT-EXCHANGER-NUMBER-EDD
(defmacro heat-exchanger-number-edd (name)
  `(info-heat-exchanger-number (tube-info ,name)))

; FREQUENCY-EDD
(defmacro frequency-edd (name)
  `(info-frequency (tube-info ,name)))

; DATE-EDD
(defmacro date-edd (name)
  `(info-date (tube-info ,name)))

; TIME-EDD
(defmacro time-edd (name)
  `(info-time (tube-info ,name)))

; SUPERVISOR-EDD
(defmacro supervisor-edd (name)
  `(info-supervisor (tube-info ,name)))

; COMMENT-EDD
(defmacro comment-edd (name)
  `(info-comment (tube-info ,name)))


;-------------------------------------------------------------------------
;                    EVENT STRUCTURE AND PROCEDURES                       |
;-------------------------------------------------------------------------


; EVENT
; THIS STRUCTURE CONTAINS THE X AND Y DATA AND THE BASE LINE FOR EACH EVENT
; EACH SLOT IS A LIST OVER EVENTS
(defstruct (event)
  (x nil)                                    ;  X VALUES
```

```
  (x-base nil)                                    ;  X BASE LINES
  (y nil)                                         ;  Y VALUES
  (y-base nil))                                   ;  Y BASE LINES

; EVENT-FILL IS THE NAME OF THE PROCEDURE THAT FILLS EVENT.
;  IT IS CONTAINED IN EVENT.LSP
(setf (get 'event 'procedure) 'event-fill)

; X-EDD
(defmacro x-edd (name)
  `(event-x (tube-event ,name)))


; X-BASE-EDD
(defmacro x-base-edd (name)
  `(event-x-base (tube-event ,name)))

; Y-EDD
(defmacro y-edd (name)
  `(event-y (tube-event ,name)))

; Y-BASE
(defmacro y-base-edd (name)
  `(event-y-base (tube-event ,name)))

; EVENT-PARMS-EDD
;(defmacro event-parms-edd (name)
; `(event-event-parms (tube-event ,name)))


;-------------------------------------------------------------------------
;                    LIMITS STRUCTURE AND PROCEDURES                      |
;-------------------------------------------------------------------------


; LIMITS
; MAXIMUM AND MINIMUM X AND Y VALUES OVER ALL TUBE DATA
; THE VALUES HERE ARE USED FOR DISPLAY PURPOSES
; MAX-X-MIN-Y IS NO LONGER USED
(defstruct (limits)
  (max-x nil)
  (min-x nil)
  (max-y nil)
  (min-y nil)
  (max-y-min-x nil))

; LIMITS-FILL IS THE NAME OF THE PROCEDURE THAT FILLS LIMITS
;  IT IS CONTAINED IN LIMITS.LSP
```

```
(setf (get 'limits 'procedure) 'limits-fill)

; MAX-X-EDD
(defmacro max-x-edd (name)
  `(limits-max-x (tube-limits ,name)))


; MIN-X-EDD
(defmacro min-x-edd (name)
  `(limits-min-x (tube-limits ,name)))

; MAX-Y-EDD
(defmacro max-y-edd (name)
  `(limits-max-y (tube-limits ,name)))

; MIN-Y-EDD
(defmacro min-y-edd (name)
  `(limits-min-y (tube-limits ,name)))


; MAX-Y-MIN-X-EDD
(defmacro max-y-min-x-edd (name)
  `(limits-max-y-min-x (tube-limits ,name)))

; LIMITS-PARMS-EDD
;(defmacro limits-parms-edd (name)
;  `(limits-limits-parms (tube-limits ,name)))



;-------------------------------------------------------------------------
;          CALCULATED-PROPERTIES STRUCTURE AND PROCEDURES                 |
;-------------------------------------------------------------------------

; CALC
; CALCULATED PROPERTIES OF THE EVENTS
; EACH ITEM IS A LIST OVER EVENTS
; THE ITEMS HERE ARE NEEDED TO MAKE THE INITIAL DETERMINATION OF PROSPECTIVE
;  SUPPORT PLATES (PROSP)
(defstruct (calc)
  (tip-index nil)                          ; REFERS TO X,Y TUBE-DATA
  (tip-angle nil)                          ; DEGREES WRT -X AXIS
  (tip-length nil)                         ; LENGTH TO TIP
  (vector-ends-out  nil)                   ; OUT LIST OF THE
                                           ;  INDEXES (REFERS TO X,Y
                                           ;  DATA) OF THE BASIS
                                           ;  VECTOR END POINTS


  (vector-ends-back nil)
```

```
  (basis-out nil)                                   ; OUT LISTS OF
  (basis-back nil)                                  ;   BASIS VECTOR ANGLES
  (basis-length-out nil)                            ; LENGTH OF BASIS-OUT
  (basis-length-back nil)
  (initial-angle-out nil)                           ; (CAR BASIS-OUR)
  (initial-angle-back nil))

; CALCULATED-PROPERTIES-FILL IS THE NAME OF THE PROCEDURE THAT FILLS
;  CALCULATED-PROPERTIES.  IT IS CONTAINED IN CALCPROP.LSP
(setf (get 'calculated-properties 'procedure) 'calculated-properties-fill)

; TIP-INDEX-EDD
(defmacro tip-index-edd (name)
  `(calc-tip-index (tube-calculated-properties ,name)))

; TIP-ANGLE-EDD
(defmacro tip-angle-edd (name)
  `(calc-tip-angle (tube-calculated-properties ,name)))

; TIP-LENGTH-EDD
(defmacro tip-length-edd (name)
  `(calc-tip-length (tube-calculated-properties ,name)))

; VECTOR-ENDS-OUT-EDD
(defmacro vector-ends-out-edd (name)
  `(calc-vector-ends-out (tube-calculated-properties ,name)))

; VECTOR-ENDS-BACK-EDD
(defmacro vector-ends-back-edd (name)
  `(calc-vector-ends-back (tube-calculated-properties ,name)))

; BASIS-OUT-EDD
(defmacro basis-out-edd (name)
  `(calc-basis-out (tube-calculated-properties ,name)))

; BASIS-BACK-EDD
(defmacro basis-back-edd (name)
  `(calc-basis-back (tube-calculated-properties ,name)))

; BASIS-LENGTH-OUT-EDD
(defmacro basis-length-out-edd (name)
  `(calc-basis-length-out (tube-calculated-properties ,name)))

; BASIS-LENGTH-BACK-EDD
(defmacro basis-length-back-edd (name)
  `(calc-basis-length-back (tube-calculated-properties ,name)))

; INITIAL-ANGLE-OUT-EDD
```

```
(defmacro initial-angle-out-edd (name)
  `(calc-initial-angle-out (tube-calculated-properties ,name)))

; INITIAL-ANGLE-BACK-EDD
(defmacro initial-angle-back-edd (name)
  `(calc-initial-angle-back (tube-calculated-properties ,name)))



;-------------------------------------------------------------------------
;                    NOSP STRUCTURE AND PROCEDURES                        |
;-------------------------------------------------------------------------

; NOSP
; LIST OVER FEATURES (EXCEPT AVERAGES)
; CONTAINS PROPERTIES NECESSARY TO DETERMINE PROSPECTIVE SUPPORT PLATES
;  (PROSP), NORMAL SUPPORT PLATES (NOSP) AND AVERAGES OVER PROSPS AND NOSPS
(defstruct (nosp)
  (s-shape-out? nil)                          ; F OR INDEX
  (s-shape-back? nil)
  (out=back? nil)                             ; T OR F
  (prosp? nil)                                ; T OR NIL
  (prosp-avg-tip-length nil)
  (prosp-avg-tip-angle nil)
  (nosp? nil)                                 ; T OR NIL
  (nosp-events nil)                           ; LIST OF NOSP EVENTS
  (nosp-avg-tip-length nil)                   ; AVERAGE OVER NOSPS
  (nosp-avg-tip-angle nil))                   ; AVERAGE OVER NOSPS

; NOSP-FILL IS THE NAME OF THE PROCEDURE THAT FILLS NOSP
;  IT IS CONTAINED IN NOSP.LSP
(setf (get 'nosp 'procedure) 'nosp-fill)

; S-SHAPE-OUT?-EDD
(defmacro s-shape-out?-edd (name)
  `(nosp-s-shape-out? (tube-nosp ,name)))

; S-SHAPE-BACK?-EDD
(defmacro s-shape-back?-edd (name)
  `(nosp-s-shape-back? (tube-nosp ,name)))

; OUT=BACK?-EDD
(defmacro out=back?-edd (name)
  `(nosp-out=back? (tube-nosp ,name)))

; PROSP?-EDD
(defmacro prosp?-edd (name)
```

```
  '(nosp-prosp? (tube-nosp ,name)))

; PROSP-AVG-TIP-LENGTH-EDD
(defmacro prosp-avg-tip-length-edd (name)
  '(nosp-prosp-avg-tip-length (tube-nosp ,name)))

; PROSP-AVG-TIP-ANGLE-EDD
(defmacro prosp-avg-tip-angle-edd (name)
  '(nosp-prosp-avg-tip-angle (tube-nosp ,name)))

; NOSP?-EDD
(defmacro nosp?-edd (name)
  '(nosp-nosp? (tube-nosp ,name)))

; NOSP-EVENTS-EDD
(defmacro nosp-events-edd (name)
  '(nosp-nosp-events (tube-nosp ,name)))

; NOSP-AVG-TIP-LENGTH-EDD
(defmacro nosp-avg-tip-length-edd (name)
  '(nosp-nosp-avg-tip-length (tube-nosp ,name)))

; NOSP-AVG-TIP-ANGLE-EDD
(defmacro nosp-avg-tip-angle-edd (name)
  '(nosp-nosp-avg-tip-angle (tube-nosp ,name)))


;-------------------------------------------------------------------------
;                    NON-NOSP STRUCTURE AND PROCEDURES                    |
;-------------------------------------------------------------------------

; NON-NOSP
; FEATURES OF NON-NOSP EVENTS
; LIST OVER EVENTS (NIL IF NA)
; BASIS VECTORS ETC ARE RECALCULATED FOR MORE RESOLUTION FOR THE FAULTS IE
;  NON-NOSP EVENTS
(defstruct (non-nosp)
  (nn-ve-out nil)                          ; NON-NOSP-VECTOR-ENDS-OUT
  (nn-ve-back nil)                         ; NON-NOSP-VECTOR-ENDS-BACK
  (nn-b-out nil)                           ; NON-NOSP-BASIS-OUT
  (nn-b-back nil)                          ; NON-NOSP-BASIS-BACK
  (nn-bl-out nil)                          ; NON-NOSP-BASIS-LENGTH-OUT
  (nn-bl-back nil)                         ; NON-NOSP-BASIS-LENGTH-BACK
  (nn-ia-out nil)                          ; NON-NOSP-INITIAL-ANGLE-OUT
  (nn-ia-back nil)                         ; NON-NOSP-INITIAL-ANGLE-BACK
  (nn-o=b? nil)                            ; NON-NOSP-OUT=BACK?
  (curve-to--x-out? nil)
```

```
   (curve-to--x-back? nil)
   (wall-loss nil))

; NON-NOSP-FILL IS THE NAME OF THE PROCEDURE THAT FILLS NON-NOSP
;  IT IS CONTAINED IN NON-NOSP.LSP
(setf (get 'non-nosp 'procedure) 'non-nosp-fill)

; NN-VE-OUT-EDD
(defmacro nn-ve-out-edd (name)
  `(non-nosp-nn-ve-out (tube-non-nosp ,name)))

; NN-VE-BACK-EDD
(defmacro nn-ve-back-edd (name)
  `(non-nosp-nn-ve-back (tube-non-nosp ,name)))

; NN-B-OUT-EDD
(defmacro nn-b-out-edd (name)
  `(non-nosp-nn-b-out (tube-non-nosp ,name)))

; NN-B-BACK-EDD
(defmacro nn-b-back-edd (name)
  `(non-nosp-nn-b-back (tube-non-nosp ,name)))

; NN-BL-OUT-EDD
(defmacro nn-bl-out-edd (name)
  `(non-nosp-nn-bl-out (tube-non-nosp ,name)))

; NN-BL-BACK-EDD
(defmacro nn-bl-back-edd (name)
  `(non-nosp-nn-bl-back (tube-non-nosp ,name)))

; NN-IA-OUT-EDD
(defmacro nn-ia-out-edd (name)
  `(non-nosp-nn-ia-out (tube-non-nosp ,name)))

; NN-IA-BACK-EDD
(defmacro nn-ia-back-edd (name)
  `(non-nosp-nn-ia-back (tube-non-nosp ,name)))

; NN-O=B?-EDD
(defmacro nn-o=b?-edd (name)
  `(non-nosp-nn-o=b? (tube-non-nosp ,name)))

; CURVE-TO--X-OUT?-EDD
(defmacro curve-to--x-out?-edd (name)
  `(non-nosp-curve-to--x-out? (tube-non-nosp ,name)))

; CURVE-TO--X-BACK?-EDD
```

```
(defmacro curve-to--x-back?-edd (name)
  `(non-nosp-curve-to--x-back? (tube-non-nosp ,name)))

; WALL-LOSS-EDD
(defmacro wall-loss-edd (name)
  `(non-nosp-wall-loss (tube-non-nosp ,name)))



;---------------------------------------------------------------------------
;                    ANALYSIS STRUCTURE AND PROCEDURES                      |
;---------------------------------------------------------------------------

; ANALYSIS
; LIST OVER EVENTS
; EACH ITEM OF LIST CONTAINS EITHER "sp" (INDICATING THE RESPECTIVE EVENT IS
;  A SUPPORT PLATE ), NIL (INDICATING THAT THE PROGRAM COULD NOT ANALYZE THE
;  EVENT) OR A LIST OF POSSIBLE ANALYSIS
(defstruct (analysis)
  (list nil))

; ANALYSIS-FILL IS THE NAME OF THE PROCEDURE THAT FILLS ANALYSIS
;   IT IS CONTAINED IN ANALYSIS.LSP
(setf (get 'analysis 'procedure) 'analysis-fill)

; LIST-EDD
(defmacro list-edd (name)
  `(analysis-list (tube-analysis ,name)))

; ANALYSIS-PARMS-EDD
;(defmacro analysis-parms-edd (name)
;  `(analysis-analysis-parms (tube-analysis ,name)))

;---------------------------------------------------------------------------
;                       TUBE STRUCTURE AND PROCEDURE                        |
;---------------------------------------------------------------------------


; TUBE
; THERE ARE NO PROCEDURES TO GET-VALUE OR PUT-VALUE PARAMETERS ETC
;   BECAUSE THIS MIGHT RESULT IN A CHANGE OF THE STRUCTURE OF TUBE
(defstruct (tube)
  (slot-list '(parameters info event limits     ; LIST OF OTHER SLOTS
               calculated-properties
               nosp non-nosp analysis))
  (parameters (make-parameters))                ; PARAMETERS
  (info (make-info))                            ; TUBE INFO LIKE STATION ETC
  (event (make-event))                          ; SEPARATED EVENTS
```

```
  (limits (make-limits))                    ; LIMITS OF X AND Y OVER TUBE
  (calculated-properties (make-calc))       ; CALCULATED PROPERTIES
  (nosp (make-nosp))                        ; NOSP FEATURES
  (non-nosp (make-non-nosp))                ; NON-NOSP PROPERTIES
  (analysis (make-analysis)))               ; EVENT ANALYSIS

; SLOT-LIST-EDD
(defmacro slot-list-edd (name)
  `(tube-slot-list ,name))
```

```
; THIS FILE CONTAINS THE PROCEDURES FOR PLACING IN RAM A FILLED TUBE STRUCTURE
; THE MAIN PROCEDURE IS FRONT-AND -CENTER

; FRONT-AND-CENTER
; RETURNS T AND FILLS A STRUCTURE WITH NAME NAME IF RAW TUBE DATA IS ON DISK;
;   OTHERWISE RETURNS NIL
; FRONT-AND-CENTER WILL (RE)CALCULATE STARTING WITH SLOT SLOT (THE DEFAULT,
; NIL, CAUSES FRONT-AND-CENTER TO FIND (LOAD FROM NAME.S - IF NECESSARY) AND
;   USE AS MUCH OF THE EXISTING STRUCTURE AS EXISTS)
; IN ANY CASE IF FRONT-AND-CENTER IS TOLD TO USE THE EXISTING STRUCTURE AND
;   IT IS NOT IN RAM THEN IT LOOKS ON THE DISK.  IF IT IS NOT THERE THE
;   WHOLE STRUCTURE IS CALCULATED.
; IF FRONT-AND-CENTER IS TOLD TO USE AN EXISTING PART OF THE STRUCTURE AND IT
;   DOESN'T EXIST THEN IT IS CALCULATED.

(defun front-and-center (name                          ; NAME CONTAINS THE STRUCTURE
                        &optional (slot nil))          ;  LABEL (NOT THE STRUCTURE)
  (cond ((raw-data? name)                              ; RAW DATA EXISTS
         (cond ((tube-in-ram? name))                   ; STRUCTURE NAME IN RAM
               ((read-tube.s name))                    ; STRUCTURE NAME ON DISK
               (T (set-name name)))                    ; NEW STRUCTURE
         (fill-slots                                   ; FROM SLOT OR FIRST NIL SLOT
            name (earlier                              ; EARLIER SLOT
                  name slot                            ; NIL => END
                    (find-first-nil name)))            ; FIRST NIL SLOT IN NAME
         (save-tube name))                             ; SAVE NAME ON DISK
        (T nil)))                                      ; RAW DATA DOESN'T EXIST


; RAW-DATA?
; CHECKS TO SEE IF THE DATA FILE EXISTS ON DISK
(defun raw-data? (name)
  (data)                                               ; SET DOS DIRECTORY
  (probe-file (symbol-name name)))

; TUBE-IN-RAM?
; CHECKS TO SEE IF THE STRUCTURE EXISTS IN RAM
(defun tube-in-ram? (name)
  (ignore-errors (named-structure-p (eval name))))

; READ-TUBE.S
; READS TUBE.S STRUCTURE NAME.S FROM DISK
; IF IT DOESN'T EXIST THEN NIL IS RETURNED
(defun read-tube.s (name)
  (data)
  (if (probe-file (setf file
                        (string-append "d:\\lsp-2\\eddy\\data\\"
                                (symbol-name name) ".s")))
```

```
      (load file)
      nil))


; SET-NAME
; SETS NAME TO A BLANK TUBE STRUCTURE
(defun set-name (name)
  (setf (symbol-value name) (make-tube)))

; FILL-SLOTS
; FILLS THE SLOTS OF NAME ON SLOT-LIST
(defun fill-slots (name slot-list)
  (parameters-fill name)                          ; ALWAYS FILL THE PARAMETERS
                                                  ;  SLOT SO THAT YOU CAN FILL
                                                  ;  ANY SLOT, USING NEW
                                                  ;  PARAMETERS
  (dotimes (i (length slot-list))
       (let* ((slot (nth i slot-list))
              (procedure (get-procedure slot)))
        (funcall procedure name))))

; EARLIER
; RETURNS THE NAME OF THE EARLIER OF SLOT AND FIRST UNFILLED STRUCTURE IN NAME
;    - THE PARAMETERS SLOT BEING INFO AND THE LATEST BEING ANALYSIS
(defun earlier (name slot first-nil)
  (let ((structure (eval name)))
    (longer (member slot (get-value structure slot-list))
            (member first-nil (get-value structure slot-list)))))

; LONGER
; RETURNS THE LONGER OF TWO LISTS
(defun longer (list-1 list-2)
  (if (> (length list-1) (length list-2))
      list-1
      list-2))

; FIND-FIRST-NIL
; FINDS THE FIRST UNFILLED STRUCTURE IN NAME
(defun find-first-nil (name)
  (let ((structure (eval name)))
    (cond ((not (get-value structure or-trigger)) 'parameters)
          ((not (get-value structure tube-number)) 'info)
          ((not (get-value structure x)) 'event)
          ((not (get-value structure max-x)) 'limits)
          ((not (get-value structure tip-index)) 'calculated-properties)
          ((not (get-value structure s-shape-out?)) 'nosp)
          ((not (get-value structure curve-to--x-out?)) 'non-nosp)
          (T nil))))                              ; ALL SLOTS FILLED
```

```
; SAVE-TUBE
; SAVES THE TUBE STRUCTURE NAME IN NAME.S
;  IN THE FORMAT (SETF NAME '(...)) SO THAT IF THE FILE NAME.S IS LOADED,
;  THE TUBE STRUCTURE WILL HAVE LABEL NAME
(defun save-tube (name)
  (data)
  (setf stream (open (pathname (string-append (symbol-name name) ".S"))
                     :direction ':output))
  (princ "(setf " stream)
  (princ (symbol-name name) stream)
  (terpri stream)
  (princ (eval name) stream)
  (princ ")" stream)
  (close stream)
  T)
```

```
; PARAMETERS-FILL
; THESE PARAMETERS CAN BE CHANGED AND THE ANALYSIS RERUN WITHOUT RECOMPILING
; THE FIRST LINE GIVES THE FILE WHERE THE CONSTANT IS USED AND THE SECOND
;  LINE GIVES THE PROCEDURE
(defun parameters-fill (name)
  (let ((structure (eval name)))
    (put-value structure or-trigger 5000)        ;USED IN EVENT.LSP
                                                 ;  CAPTURE-EVENT
    (put-value structure and-trigger 250)        ; USED IN EVENT.LSP
                                                 ;  CAPTURE-EVENT
    (put-value structure or-minimum 25)          ; USED IN EVENT.LSP
                                                 ;  FIND-TRIGGER-OR
    (put-value structure and-minimum 5)          ; USED IN EVENT.LSP
                                                 ;  FIND-TRIGGER-AND
    (put-value structure and-minimum-length 5)   ; USED IN EVENT.LSP
                                                 ;  FIND-TRIGGER-AND
    (put-value structure angles 64)              ; USED IN CALCPROP.LSP
                                                 ;  CALCULATED-PROPERTIES-FILL
    (put-value structure tip .9)                 ; USED IN CALCPROP.LSP
                                                 ;  FIND-TIP
    (put-value structure basis-length .1)        ; USED IN CALCPROP.LSP
                                                 ;  CALCULATED-PROPERTIES-FILL
    (put-value structure length-comp .2)         ; USED IN NOSP.LSP
                                                 ;  NOSP-FILL
                                                 ; ALSO IN NON-NOSP.LSP
    (put-value structure diff-vect 1.)           ; USED IN NOSP.LSP
                                                 ;  NOSP-FILL
                                                 ; ALSO IN NON-NOSP.LSP
    (put-value structure nosp-length .1)         ; USED IN NOSP-ID.LSP
                                                 ;  PROSP-AVG-LEN
    (put-value structure nosp-angle 5)           ; USED IN NOSP-ID.LSP
                                                 ;  PROSP-AVG-ANGLE
    (put-value structure non-n-basis-length      ; USED IN NON-NOSP.LSP -
                        9)))                     ;  FILL-NON-NOSP-SLOTS
```

```
; INFO-FILL
; READS THE HEADER INFORMATION FROM THE ANDES TUBE DATA FILE AND PUTS IT
;  INTO THE  TUBE STRUCTURE
(defun info-fill (name)
    (data)                                          ; CHANGE DOS DIRECTORIES
    (setf stream-file (open (pathname (symbol-name name))
                            :direction ':input
                            :element-type 'unsigned-byte))
  (let* ((structure (eval name)))
    (setf info (read-header 184))                   ; ALL INFO IN 1ST 184 CHARS
    (put-value structure tube-number (string-get 1 12 info))
    (put-value structure station (string-get 13 2 info))
    (put-value structure unit (string-get 15 12 info))
    (put-value structure heat-exchanger-type (string-get 27 24 info))
    (put-value structure heat-exchanger-number (string-get 51 25 info))
    (put-value structure frequency (string-get 76 8 info))
    (put-value structure date (string-get 84 4 info))
    (put-value structure time (string-get 88 2 info))
    (put-value structure supervisor (string-get 90 8 info))
    (put-value structure comment (string-get 98 82 info))
    (close stream-file)))

; STRING-GET
; TAKES A LIST OF CHARACTER NUMBERS AND RETURNS AS A STRING THE NUMBER
;  CHARACTERS STARTING AT ELEMENT START.
; NON-KEYBOARD SYMBOLS AND BLANKS FROM THE END ARE DISCARDED
(defun string-get (start number list-in)
    (setf list nil)
    (dotimes (i number)
      (setf list (cons (nth (+ start i) list-in) list)))
    (setf list (reverse list))
    (catch 'up (dotimes (i (length list))           ; THROW AWAY NON-KEYBOARD
                                                    ;  CHARACTERS FROM END
                 (when (or (> (nth i list) 126)
                           (< (nth i list) 32))
                       (setf list (butlast list (- (length list) i)))
                       (throw 'up nil))))
    (catch 'up (dotimes (i (length list))           ; THROW AWAY BLANKS FROM END
                 (if (equal (last list) 32)
                     (setf list (butlast list 1))
                     (throw 'up nil))))
    (setf list (mapcar #'string list))              ; CHARACTERS
    (setf string "")
    (dotimes (i (length list))                      ; CONCATENATE
      (setf string (string-append string (nth i list))))
    string)

; READ-HEADER
```

```
; READ 1ST 184 CHARACTER VALUES (CHECKING FOR END OF FILE) AND RETURN
;  AS A LIST
(defun read-header (n)
  (setf x nil)
  (catch 'up (dotimes (i n)
               (setf x-value (read-char stream-file nil 2048))
               (when (= x-value 2048) (setf end-file t)
                                      (throw 'up nil)) ;end of file
               (setf x (cons x-value x))))
  (reverse x))
```

```
; READS IN RAW TUBE DATA IN ANDES FORMAT FROM FILE NAME,
;  CONVERTS FROM 2's COMPLEMENT  TO DECIMAL, SUBTRACTS THE BASELINE, AND
;  PLACES THE X&Y AND BASELINE VALUES IN TUBE DATA STRUCTURE NAME.

; EVENT-FILL
; FILLS THE EVENT STRUCTURE IN TUBE
(defun event-fill (name)
  (data)                                          ; CHANGE DOS DIRECTORIES
  (setf length-file (read-length-file name))      ; USES DOS DIR TO READ
                                                  ; FILE LENGTH (NOT USED)
  (setf stream-file (open (pathname (symbol-name name))
                          :direction ':input
                          :element-type 'unsigned-byte))
  (setf base-x 0 base-y 0)
  (read-data 92)                                  ; FIRST 184 CHARS ARE TUBE
                                                  ;  INFO - SEE INFO.LSP
                                                  ;  READ-DATA READS CHARS IN
                                                  ;  PAIRS
  (read-convert-base 100)                         ; NEXT 100 NUMBERS SET THE
                                                  ;  INITIAL X & Y BASE LINES
  (setf base-x (round (average x))
        base-y (round (average y)))
  (let ((name-struc (eval name)))
    (zero-values name-struc)
    (catch 'up (loop (capture-event name-struc)   ; LOOK FOR A PEAK IN THE X&Y
                     (when end? (throw 'up nil))  ;  DATA
                     (print-x&y name-struc)))     ; PUT DATA IN TUBE STRUCTURE
    (reverse-values name-struc))                  ; VALUES ARE IN REVERSED
                                                  ;  ORDER
  (close-all-files))


; ZERO-VALUES
; OVERWRITES THE EVENT SLOT OF TUBE WITH DEFAULT VALUES
(defun zero-values (name-struc)
  (setf (tube-event name-struc) (make-event)))


; READ-DATA
; READS A FILE CHARACTER BY CHARACTER CHECKING FOR THE END OF FILE
; 2N CHARACTERS ARE READ
(defun read-data (n)
  (setf x nil y nil end-file nil)
  (catch 'up (dotimes (i n)
               (setf x-value (read-char stream-file nil 2048))
               (when (= x-value 2048) (setf end-file t)
                                      (throw 'up nil)) ;end of file
```

```
            (setf y-value (read-char stream-file nil 2048))
            (when (= y-value 2048) (setf end-file t)
                                   (throw 'up nil))
            (setf x (cons x-value x) y (cons y-value y))))
  (setf x (reverse x) y (reverse y)))

; READ-CONVERT-BASE
; READS 2N CHARACTERS, CONVERTS TO DECIMAL AND SUBTRACTS THE BASELINE
(defun read-convert-base (n)
  (read-data n)
  (setf x (convert-data x)                          ; X&Y ARE GLOBAL
        y (convert-data y))
  (setf x (sub-base x base-x)                       ; BASE-X & ..Y ARE GLOBAL
        y (sub-base y base-y)))

; CONVERT-DATA
; CONVERTS FROM 2's COMPLEMENT TO DECIMAL INTEGERS
(defun convert-data (x)
  (mapcar #'(lambda (y) (cond ((> y 127) (- y 256))
                              (T y)))
          x))

; SUB-BASE
; SUBTRACTS THE BASELINE FROM EACH ELEMENT IN THE LIST BASE
(defun sub-base (z base)
  (mapcar #'(lambda (element) (- element base)) z))

; CAPTURE-EVENT
; FINDS A PEAK IN THE DATA
; BEGINNING OF PEAK: EITHER X OR Y VALUE (MINUS BASE LINE) > OR-MINIMUM
; END OF PEAK: BOTH X AND Y VALUE (MINUS BASE LINE) < AND-MINIMUM
(defun capture-event (structure)
  (setf and-trigger (get-value structure and-trigger)
        or-trigger (get-value structure or-trigger)
        or-minimum (get-value structure or-minimum)
        and-minimum (get-value structure and-minimum)
        and-minimum-length (get-value structure and-minimum-length))
  (catch 'up (loop (find-trigger-or or-trigger      ; LOOK FOR X OR Y >
                                    or-minimum)     ;  OR-MINIMUM
                   (when (or end?
                             (not find-or-trigger?))
                         (throw 'up nil))
                   (setf x1 (last-n x 20)           ; SAVE LAST 20 ELEMENTS
                         y1 (last-n y 20))          ;  FOR FRONT OF EVENT
                   (find-trigger-and                ; LOOK FOR X AND Y <
                                     and-trigger    ;  AND-MINIMUM
                                     and-minimum and-minimum-length)
                   (when end?
```

```
                    (throw 'up nil))
               (when find-and-trigger?
                    (setf event-x (append x1 x) event-y (append y1 y))
                    (read-convert-base 30)    ;
                    (new-base (last-n x 10)
                              (last-n y 10))  ; CALC NEW BASE
                    (setf event-x (append     ; ADD 1ST 15 TO END OF EVENT
                                          event-x (first-n x 15))
                          event-y (append event-y (first-n y 15)))
                    (setf find-event? T)
                    (throw 'up nil))
               (when (> (length x) 9)              ; NO EVENT FOUND =>
                                                   ;  BASE LINE IS CHANGING
                    (new-base (last-n x 10) (last-n y 10))))))

; FIND-TRIGGER-OR
; SEARCHES UNTIL EITHER X OR Y (MINUS THE CURRENT BASE) ARE > MINIMUM
;  - THIS MAY INDICATE THE BEGINNING OF AN EVENT
; CALLED BY CAPTURE-EVENT
(defun find-trigger-or (trigger minimum)
  (setf x nil y nil)
  (setf end? nil)
  (catch 'up (loop
               (setf x-value (- (convert-num (read-char stream-file nil 2304))
                                base-x)) ;convert-num subtracts 256 from 2304
               (when (= x-value (- 2048 base-x))
                    (setf end? T) (throw 'up nil))
               (setf y-value (- (convert-num (read-char stream-file nil 2304))
                                base-y))
               (when (= y-value (- 2048 base-y))
                    (setf end? T) (throw 'up nil))
               (setf x (cons x-value x) y (cons y-value y))
               (when (> (length x) trigger)        ; START AGAIN
                    (setf find-or-trigger? nil)
                    (throw 'up nil))
               (when (or (> (abs x-value) minimum) ; START OF EVENT
                         (> (abs y-value) minimum))
                    (setf find-or-trigger? T)
                    (setf x (reverse x) y (reverse y))
                    (throw 'up nil)))))

; LAST-N
; RETURNS THE LAST N ELEMENTS OF A LIST OR THE LIST IF THE LENGTH IS < N
; CALLED BY CAPTURE-EVENT
(defun last-n (x n)
  (cond ((> (length x) n) (nthcdr (- (length x) n) x))
        (T x)))
```

```
; FIRST-N
; RETURNS THE FIRST N ELEMENTS OF A LIST OR THE LIST IF THE LENGTH IS < N
; CALLED BY CAPTURE-EVENT
(defun first-n (x n)
  (cond ((> (length x) n) (butlast x (- (length x) n)))
        (T x)))

; FIND-TRIGGER-AND
; SEARCHES UNTIL BOTH X&Y (MINUS THE CURRENT BASE) ARE < MINIMUM
;  - THIS MARKS THE END OF AN EVENT
; EVENT IS IGNORED IF IT IS SHORTER THAN MINIMUM-LENGTH OR LONGER THAN TRIGGER
; CALLED BY CAPTURE-EVENT
(defun find-trigger-and (trigger minimum minimum-length)
  (setf x nil y nil)
  (setf end? nil)
  (catch 'up (loop
               (setf x-value (- (convert-num (read-char stream-file nil 2304))
                                base-x))
               (when (= x-value (- 2048 base-x))
                     (setf end? T) (throw 'up nil))
               (setf y-value (- (convert-num (read-char stream-file nil 2304))
                                base-y))
               (when (= y-value (- 2048 base-y))
                     (setf end? T) (throw 'up nil))
               (setf x (cons x-value x) y (cons y-value y))
               (when (> (length x) trigger)
                     (setf find-and-trigger? nil)
                     (throw 'up nil))
               (when (and (< (abs x-value) minimum); END OF EVENT
                          (< (abs y-value) minimum))
                     (when (< (length x)           ; TOO SHORT
                              minimum-length)
                           (setf find-and-trigger? nil)
                           (throw 'up nil))
                     (setf find-and-trigger? T)
                     (setf x (reverse x) y (reverse y))
                     (throw 'up nil)))))

; NEW-BASE
; NEW BASELINE VALUES
; CALLED BY CAPTURE-EVENT
(defun new-base (x-list y-list)
  (setf base-x (+ (round (average x-list)) base-x)
        base-y (+ (round (average y-list)) base-y)))

; CONVERT-NUM
; CONVERTS A NUMBER FROM 2's COMPLEMENT TO DECIMAL
```

```
(defun convert-num (x)
  (cond ((> x 127) (- x 256))
        (T x)))

; READ-LENGTH-FILE
; READS THE LENGTH OF A FILE ASSUMING DOS IS IN THE PROPER DIRECTORY
; NOT USED
(defun read-length-file (name)
  (setf char-bag-1 (string-append "abcdefghijklmnopqrstuvwxyz"
                                  "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                                  "0123456789"
                                  "!@#$%^&()-_{}~`'"
                                  "."))    ;anything found in a dos file name
  (sys:dos (string-append "dir " (symbol-name name) ">length.fle"))
  (setf path (pathname "length.fle"))
  (setf stream (open path :direction ':input))
  (dotimes (i 4) (read-line stream))
  (setf file-line (read-line stream))
  (setf file-length (string-left-trim " "
                      (string-left-trim char-bag-1 file-line)))
  (setf file-length                                    ; STRING-RIGHT-TRIM DOESN'T
                                                       ;  SEEM TO WORK
        (car (multiple-value-list (read-from-string file-length :end " "))))
  (close stream)
  (sys:dos "erase length.fle")
  (sys::8087-fpp :use))

; AVERAGE
; AVERAGE OF THE ELEMENTS OF Z
(defun average (z)
  (/ (apply #'+ z) (length z)))                    ; MAX (LENGTH Z) = 127

; PRINT-X&Y
; PRINTS THE X&Y AND BASELINE DATA IN GLOBAL VARIABLES X, BASE-X, Y,
;  AND BASE-Y TO THE TUBE STRUCTURE.  THE DATA ENDS UP IN REVERSE ORDER.
(defun print-x&y (name-struc)
  (let ((list (cons event-x (get-value name-struc x))))
    (put-value name-struc x list ))
  (let ((list (cons base-x (get-value name-struc x-base))))
    (put-value name-struc x-base list ))
  (let ((list (cons event-y (get-value name-struc y))))
    (put-value name-struc y list ))
  (let ((list (cons base-y (get-value name-struc y-base))))
    (put-value name-struc y-base list )))


; REVERSE-VALUES
; REVERSES X, Y, X-BASE AND Y-BASE LISTS IN TUBE STRUCTURE SINCE THEY WERE
```

```
;  PUT IN IN REVERSE ORDER
(defun reverse-values (name-struc)
  (let ((list (reverse (get-value name-struc x))))
    (put-value name-struc x list))
  (let ((list (reverse (get-value name-struc x-base))))
    (put-value name-struc x-base list))
  (let ((list (reverse (get-value name-struc y))))
    (put-value name-struc y list))
  (let ((list (reverse (get-value name-struc y-base))))
    (put-value name-struc y-base list)))
```

```
; THIS FILE CONTAINS THE PROCEDURES TO FILL THE LIMITS STRUCTURE

; LIMITS-FILL
; MAX-X IS THE LARGEST X VALUE (ALL EVENTS) - SIMILARLY FOR MIN-X ETC
; THESE VALUES ARE USED FOR SCALING
(defun limits-fill (name)
  (data)
  (let ((name-struc (eval name)))
    (let* ((max-x (extremum-of-list                ; MAXIMUM OVER ALL EVENTS
                    'max  (extremum-list           ; RETURNS A LIST OVER EVENTS
                                                   ;  OF THE MAXIMUM VALUES
                            'max (setf x-list      ; X-LIST IS A LIST OVER
                                                   ;  EVENTS OF THE X LISTS
                                       (get-value name-struc x)))))
           (max-y (extremum-of-list
                    'max  (extremum-list
                            'max (setf y-list (get-value name-struc y)))))
           (min-x (extremum-of-list
                    'min  (extremum-list
                            'min x-list)))
           (min-y (extremum-of-list
                    'min  (extremum-list
                            'min y-list))))
      (put-value name-struc max-x max-x)
      (put-value name-struc min-x min-x)
      (put-value name-struc max-y max-y)
      (put-value name-struc min-y min-y))))

; EXTREMUM-LIST
; M IS EITHER #'MAX OR #'MIN
; A LIST OF M (IE MAX OR MIN) VALUES OF EACH OF THE SUBLISTS OF Y IS RETURNED
;  IE Y IS OF THE FORM ((...) (...) .... (...))
(defun extremum-list (m y)
  (mapcar #'(lambda (x) (extremum-of-list m x)) y))

; EXTREMUM-OF-LIST
; M IS EITHER #'MAX OR #'MIN
; RETURNS THE M (IE MAX OR MIN) OF LIST X
; GCLISP LM 2.2 HAS A LIMIT ON APPLY OF 127 => THE RECURSIVE CALL FOR LISTS
;  WITH LENGTH > 127
(defun extremum-of-list (m x)
  (cond ((< (length x) 128) (apply m x))
        (T (funcall m (apply m (butlast x (- (length x) 127)))
                      (extremum-of-list m (nthcdr 127 x))))))
```

```
; THIS FILE CONTAINS THE PROCEDURES NEEDED TO FILL THE CALCULATED-PROPERTIES
;  STRUCTURE

; CALCULATED-PROPERTIES-FILL
; THE TRACES GENERALLY RETURN TO THE ORIGN WITH A DEFINITE TIP AT THE EXTREME.
; OUT REFERS TO THE X AND Y VALUES GOING FROM THE BALANCE POINT TO THE TIP
;  AND BACK REFERS TO THE VALUES RETURNING FROM THE TIP TO THE BALANCE POINT
; THE BASIS VECTORS IN WHICH THE OUT AND BACK TRACES ARE EXPANDED HAVE
;  A GIVEN LENGTH RELATIVE TO MIN-X AND ANGLES WHICH ARE MULTIPLES OF
;  (/ 360 (GET-VALUE STRUCTURE ANGLES))
(defun calculated-properties-fill (name)
  (let* ((structure (eval name))
         (x-list (get-value structure x))
         (y-list (get-value structure y)))
    (setf tip-index nil
          tip-angle nil
          tip-length nil
          vector-ends-out nil
          vector-ends-back nil
          basis-out nil
          basis-back nil
          basis-length-out nil
          basis-length-back nil
          initial-angle-out nil
          initial-angle-back nil)
    (dotimes (i (length x-list))                    ; DO OVER EVENTS
        (setf tip-index (cons (find-tip             ; TIP INDEX IS THE ELEMENT
                                                    ;  NUMBER OF X THAT MARKS THE
                                                    ;  THE TIP
                                structure (setf x (nth i x-list))
                                          (setf y (nth i y-list)))
                              tip-index))
        (setf tip-angle (cons (find-angle           ; ANGLE TO TIP
                                (car tip-index) x y) tip-angle))
        (setf tip-length (cons (find-length         ; LENGTH TO TIP
                                 (car tip-index) x y) tip-length))
        (setf basis-length (expt (round             ; THE EVENT TRACE IS EXPANDED
                                                    ;  IN TERMS OF BASIS VECTORS
                                                    ;  OF LENGTH BASIS-LENGTH
                             (* (get-value          ; RELATIVE TO MIN-X, MEASURES
                                                    ;  SUPPORT PLATE SIGNAL SIZE
                                structure min-x)
                                (get-value structure basis-length)))
                           2))
        (setf vector-ends-out                       ; LIST OF INDICES (OF X) OF
                                                    ;  ENDS OF THE OUT SIGNAL
                                                    ;  BASIS VECTORS
              (cons (find-string x y basis-length (car tip-index))
```

```
                    vector-ends-out))
          (setf vector-ends-back
                (cons (complement (length x)
                                  (find-string (reverse x)
                                               (reverse y)
                                               basis-length
                                               (- (length x) 1
                                                  (car tip-index))))
                    vector-ends-back))
          (setf basis-length-out (cons (length     ; NUMBER OF VECTORS
                                             (car vector-ends-out))
                                       basis-length-out))
          (setf basis-length-back (cons (length (car vector-ends-back))
                                        basis-length-back))
          (setf basis-out (cons (vect              ; ANGLES OF THE BASIS VECTORS
                                                   ;  FOR THE OUT TRACE
                                      (get-value structure angles)
                                      (calc-angles
                                        x y (car vector-ends-out)))
                                basis-out))
          (setf basis-back (cons (vect (get-value structure angles)
                                       (calc-angles
                                         x y (car vector-ends-back)))
                                 basis-back))
          (setf initial-angle-out (cons            ; ANGLE OF 1ST BASIS VECTOR
                                                   ;  FOR THE OUT TRACE
                                        (to-degrees
                                           (get-value structure angles)
                                           (caar basis-out))
                                        initial-angle-out))
          (setf initial-angle-back (cons (to-degrees
                                           (get-value structure angles)
                                           (caar basis-back))
                                         initial-angle-back)))
      (put-value structure tip-index (reverse tip-index))
      (put-value structure tip-angle (reverse tip-angle))
      (put-value structure tip-length (reverse tip-length))
      (put-value structure vector-ends-out (reverse vector-ends-out))
      (put-value structure vector-ends-back (reverse vector-ends-back))
      (put-value structure basis-out (reverse basis-out))
      (put-value structure basis-back (reverse basis-back))
      (put-value structure basis-length-out (reverse basis-length-out))
      (put-value structure basis-length-back (reverse basis-length-back))
      (put-value structure initial-angle-out (reverse initial-angle-out))
      (put-value structure initial-angle-back (reverse initial-angle-back))))

; FIND-TIP
; FINDS THE TIP OF DATA IN X & Y
```

```
; THE TIP IS DEFINED AS FOLLOWS: FIND THE MAXIMUM (+ (EXPT X 2) (EXPT Y 2))
;  VALUE FROM THE BALANCE POINT; FIND, ALONG BOTH THE OUT AND BACK TRACES, THE
;  POINT WHERE (+ (EXPT X 2) (EXPT Y 2)) EQUALS (* MAXIMUM (EXPT TIP 2)).
;  THE TIP IS DEFINED AS BEING MIDWAY (IE IN NUMBER OF X VALUES) BETWEEN
;  THESE TWO POINTS.  IT WAS DONE THIS WAY SO THAT A TRACE COULD LOOP BACK
;  ON ITSELF AND THE PROGRAM WOULD FIND THE PROPER TIP.  HOWEVER
;  A LITTLE EXPERIENCE IS SHOWING THAT IT MAKES LITTLE DIFFERENCE
;  IF THE MINIMUM X POINT WERE USED FOR DEFINING THE TIP- AND IT
;  WOULD CERTAINLY BE MUCH FASTER.
(defun find-tip (structure x y)
  (setf maxx-in 0 max 0 sq-list nil)
  (dotimes (i (length x))
    (when (> (setf new-max (+ (expt (nth i x) 2)
                              (expt (nth i y) 2)))
             max)
          (setf maxx-in i max new-max))          ; RECORD MAXIMUM
    (setf sq-list (cons new-max sq-list)))       ; RECORD LIST OF DISTANCES
                                                 ;  FROM BALANCE POINT SQUARED
  (setf sq-list (reverse sq-list))
  (let ((sq-90 (* (expt (get-value structure tip) 2) max)))
       (setf right-index                         ; FIND INDEX ON BOTH SIDES
                         (search-list sq-90 sq-list)
             left-index (- (length sq-list) 1
                           (search-list sq-90 (reverse sq-list)))))
  (setf maxx-in (round (* (+ right-index         ; TIP INDEX
                             left-index) .5)))
  maxx-in)

; SEARCH-LIST
; FINDS THE INDEX OF LIST WHERE ELEMENT >= MAX
; CALLED BY FIND-TIP
(defun search-list (max list)
  (setf imax 0)
  (catch 'up (dotimes (i (length list))
                      (when (> (nth i list) max) (setf imax i)
                                                 (throw 'up nil)))
             (setf imax (+ (length list) 1)))
  imax)

; FIND-ANGLE
; FINDS THE ANGLE FROM (0,0) TO TIP IN DEGREES WRT THE -X AXIS
; INDEXTH ELEMENT OF X IS THE TIP
; CALLED BY CALCULATED-PROPERTIES-FILL
(defun find-angle (index x y)
  (cond ((< (setf ang (round
                        (- 180
                           (* (/ 180 pi)
                              (atan (nth index y)
```

```
                                        (nth index x))))))
          180) ang)
       (T (- ang 360))))

; TIP-LENGTH
; RETURNS THE LENGTH FROM THE BALANCE POINT TO THE TIP
; INDEXTH ELEMENT OF X IS THE TIP
; CALLED BY CALCULATED-PROPERTIES-FILL
(defun find-length (index x y)
  (sqrt (+ (expt (nth index x) 2)
           (expt (nth index y) 2))))


; TO-DEGREES
; CONVERTS FROM FRACTIONS TO DEGREES WRT X AXIS
(defun to-degrees (an no) (round (* no (/ 360 an))))

; FIND-STRING
; CALLED BY CALCULATED-PROPERTIES-FILL
; FINDS THE INDICES OF X THAT REPRESENT THE END POINTS OF THE BASIS VECTORS
(defun find-string (x y basis-length max-inn)
  (setf basis-in '(0))
  (setf j 0 lenn (length x))
  (setf basis-in (list (search-index 0 0 x y lenn basis-length max-inn)))
  (catch 'up
    (loop (setf index (search-index
                        (nth (car basis-in) x)
                        (nth (car basis-in) y) x y lenn basis-length
                                               max-inn))
          (when (or (> index lenn) (> index max-inn))
                    (throw 'up nil))            ; STOP WHEN REACH END OF
                                                ;  LIST OR TIP
          (setf basis-in (cons index basis-in) j (+ j 1))))
  (setf basis-in (reverse basis-in)))

; SEARCH-INDEX
; LOOKS IN X,Y FOR NEXT VECTOR END AFTER X0,Y0
; RETURNS INDEX OF NEXT VECTOR END
; CALLED BY FIND-STRING
(defun search-index (x0 y0 x y lenn con-basis-length max-inn)
  (catch 'up (do ((i (car basis-in) (+ i 1)))
                 ((or (= i lenn) (> i max-inn)))  ; = WAS >
                 (when (> (+ (expt (- (nth i x) x0) 2)
                             (expt (- (nth i y) y0) 2))
                          con-basis-length)
                       (setf se-in i)
                       (throw 'up nil)))
             (setf se-in (+ 1 lenn)))            ; RESULTS IN VECTOR ENDS < 0
```

```
  se-in)

; COMPLEMENT
; RETURNS THE COMPLEMENT OF A LIST OF VECTOR-END INDICES
; CALLED BY BASIS-FEATURES
(defun complement (n list)
  (mapcar #'(lambda (x) (- n 1 x)) list))          ; -1 BECAUSE N = LENGTH

; CALC-ANGLES
; RETURNS A LIST OF THE ANGLES OF THE VECTORS WHOSE END INDICES ARE IN BASIS-IN
; CALLED BY BASIS-FEATURES
(defun calc-angles (x y basis-in)
  (setf angles (list
            (arctan (nth (nth 0 basis-in) x) (nth (nth 0 basis-in) y))))
  (dotimes (i (- (length basis-in) 1))
          (setf angles (cons
              (arctan (- (nth (setf j1 (nth (+ i 1) basis-in)) x)
                         (nth (setf j2 (nth i basis-in)) x))
                      (- (nth j1 y) (nth j2 y))) angles)))
  (setf angles (reverse angles)))

; ARCTAN
; RETURNS THE ANGLE WITH TAN = X/Y (IN DEGREES)
; ANGLE WRT X AXIS RANGE IS 0 TO 360
; CALLED BY CALC-ANGLES
(defun arctan (x y)
  (cond ((and (numberp x) (numberp y))
         (let ((at (* 57.29577951 (atan y x))))
           (cond ((< at 0) (+ 360 at))
                 (T at))))
        (T 0)))

; VECT
; N IS THE NUMBER OF BASIS VECTORS IN 360 DEGREES
; RETURNS LIST OF ANGLES IN ANG IN N UNITS
; CALLED BY BASIS-FEATURES
(defun vect (n ang)
  (let ((an (/ 360 n)))
       (mapcar #'(lambda (x) (round (/ x an))) ang)))
```

```
; THE PROCEDURES IN THIS FILE FILL THE NOSP SLOTS IN THE TUBE STRUCTURE

; NOSP-FILL
; THIS PROCEDURE FIRST DETERMINES WHETHER OR NOT EACH EVENT HAS AS S SHAPE
;  AS DEFINED BY THRU-PI AND WHETHER THE OUT AND BACK SIGNALS (SEPARATED BY
;  THE TIP) ARE THE SAME (SEE OUT=BACK?).  EACH EVENT IS THEN TESTED TO SEE
;  IF IT PASSES THE PROSP (PROSPECTIVE SUPPORT PLATE) RULE (SEE PRSOP.LSP)
;  AND THEN THE NOSP (NORMAL SUPPORT PLATE) RULE (SEE NOSP.LSP), BOTH
;  USING THE PROCEDURE PROCESSOR (SEE RULE.LSP).
; AFTER THE NOSP EVENTS ARE IDENTIFIED, THE AVERAGES ARE CALCULATED
(defun nosp-fill (name)
  (let* ((structure (eval name))                    ; RECALL VALUES
         (basis-out (get-value structure basis-out))
         (basis-back (get-value structure basis-back))
         (x (get-value structure x))
         (y (get-value structure y))
         (vector-ends-out (get-value structure vector-ends-out))
         (vector-ends-back (get-value structure vector-ends-back))
         (angles (get-value structure angles))
         (length-comp (get-value structure length-comp))
         (diff-vect (expt (* (get-value structure diff-vect)
                             (get-value structure min-x))
                          2)))
    (setf s-shape-out? nil                          ; INITIAL VALUES
          s-shape-back? nil
          out=back? nil
          prosp?  (make-list (length basis-out) :initial-element nil)
          nosp?  (make-list (length basis-out) :initial-element nil)
          nosp-events nil
          nosp-avg-tip-length nil
          nosp-avg-tip-angle nil
          event-analysis  (make-list (length basis-out) :initial-element nil))
    (dotimes (i (length basis-out))                 ; DO OVER EVENTS
      (setf s-shape-out? (cons (thru-pi (nth i basis-out)
                                        (trans-+-- (nth i basis-out) angles))
                               s-shape-out?))
      (setf s-shape-back? (cons (thru-pi (nth i basis-back)
                                         (trans-+-- (nth i basis-back) angles))
                                s-shape-back?))
      (setf out=back? (cons (out=back? (nth i x)
                                       (nth i y)
                                       (nth i vector-ends-out)
                                       (nth i vector-ends-back)
                                       length-comp diff-vect)
                            out=back?)))
    (put-value structure s-shape-out?               ; RECORD
                         (reverse s-shape-out?))
    (put-value structure s-shape-back? (reverse s-shape-back?))
```

```
(put-value structure out=back? (reverse out=back?))
(dotimes (i (length basis-out))                    ; RUN EACH EVENT THROUGH THE
  (setf prosp?-i (processor name 'prosp-id i)) ;  RULE PROCESSOR WITH THE
  (when prosp?-i                                   ;  RULES IN PROSP-ID.LSP
        (setf (nth i event-analysis) prosp?-i
              (nth i prosp?) T)))
(put-value structure prosp? prosp?)                ; RECORD
(put-value structure list event-analysis)
(put-value structure  prosp-avg-tip-length         ; AVERAGE TIP-LENGTHS OVER
                        (average (prosp-filter ;  THE PROSP EVENTS
                                    structure
                                    (get-value structure tip-length))))
(put-value structure prosp-avg-tip-angle           ; AVERAGE TIP-ANGLE OVER
                        (average (prosp-filter ;  THE PROSP EVENTS
                                    structure
                                    (get-value structure tip-angle))))
(dotimes (i (length basis-out))
  (setf nosp?-i (processor name 'nosp-id i))       ; RUN EACH EVENT THROUGH THE
  (when nosp?-i                                    ;  RULE PROCESSOR WITH THE
       (setf (nth i event-analysis) nosp?-i)       ;  RULE IN NOSP-ID
       (setf (nth i nosp?) T)))
(put-value structure nosp? nosp?)                  ; RECORD
(put-value structure list event-analysis)
(dotimes (i (length basis-out))                    ; LIST OF NOSP EVENTS
  (when (nth i nosp?)
        (setf nosp-events (cons i nosp-events))))
(put-value structure nosp-events (reverse nosp-events))
(put-value structure  nosp-avg-tip-length          ; AVERAGE TIP LENGTH OVER
                        (average (nosp-filter  ;  NOSPS
                                    structure
                                    (get-value structure tip-length))))
(put-value structure nosp-avg-tip-angle            ; AVERAGE TIP ANGLE OVER
                        (average (nosp-filter  ;  NOSPS
                                    structure
                                    (get-value structure tip-angle))))
))
```

```
;-------------------------------------------------------------------------
; S-SHAPE? PROGRAMS--S-SHAPE? PROGRAMS--S-SHAPE? PROGRAMS--S-SHAPE? PROGRAMS |
;-------------------------------------------------------------------------


; TRANS-+--
; TRANSFORMS A LIST OF ANGLES INTO A LIST OF + AND -
;  ACCORDING TO WHETHER THEY ARE BETWEEN PI AND 3*PI/2 (+),
;  PI/2 and PI (-) OR OTHERWISE (*)
```

```
(defun trans-+-- (angles con-angles)
  (let* ((pie (/ con-angles 2))
         (pie/2 (/ pie 2))
         (3*pie/2 (* 1.5 pie)))
    (mapcar #'(lambda (x) (cond ((and (>= x pie) (<=  x 3*pie/2)) '+)
                                ((and (>= 0) (<= x pie)) '-)
                                (T '*)))
            angles)))

; THRU-PI
; DETERMINES IF THE LIST OF ANGLES IN ANGLES HAS THE STRUCTURE:
;  - . . . - + . . . MAXIMUM (+ or -) . . . (DEFINING THE S SHAPE)
;  THE INDICES WHERE THE MAXIMUM AND INFLECTION POINT OCCUR ARE RETURNED.
;   (ALTHOUGH NEVER USED)
;  IF THE  STRUCTURE IS NOT PRESENT THEN 'F IS RETURNED.
(defun thru-pi (basis* angles*)
  (let* ((angles (butlast                          ; TAKE OUT ITEMS AFTER FIRST *
                  angles* (length (member '* angles*))))
         (len (length angles)))
    (cond ((< (length angles) 4) 'F)               ; MAKE SURE LONG ENOUGH
          ((neq (car angles) '-) 'F)               ; CHECK FIRST ITEM FOR -
          (T (setf angles (member '+ angles)       ; ALL ITEMS AFTER FIRST +
                   index-1 (- len
                              (length angles)) ; FIRST CROSSOVER INDEX
                   angles (member (apply #'max ; ALL ITEMS AFTER MAX
                                         (nthcdr (1- index-1) basis*)) basis*)
                   index-2 (cond ((>                ; INFLECTION POINT
                                   (length angles) 1)
                                  (- len (length angles)))
                                 (T 0)))
             (cond ((<= 0 index-1 index-2) (list index-1 index-2))
                   (T 'F))))))


;-------------------------------------------------------------------------
; OUT=BACK? PROGRAMS----------OUT=BACK? PROGRAMS----------OUT=BACK? PROGRAMS |
;-------------------------------------------------------------------------


; OUT=BACK?
; RETURNS T IF THE OUT AND BACK SIGNAL ARE THE SAME; OTHERWISE RETURNS 'F
(defun out=back? (x y vector-ends-out vector-ends-back length-comp diff-vect)
       (cond ((or (= (setf len-o (length           ; SHOULD'T HAPPEN
                                  vector-ends-out)) 0)
                  (= (setf len-b (length vector-ends-back)) 0)
                  (< (car vector-ends-out) 0)      ; SEE SEARCH-INDEX
                  (< (car vector-ends-back) 0)     ;  IN CALC-PROP.LSP
                  (> (car vector-ends-out)         ; IBID
```

```
                    (length x))
                (> (car vector-ends-back) (length x)))
            'F)
           ((> (abs (/ (- len-o len-b) len-o)) length-comp)
            'F)                                    ; ONE LONGER (BY NUMBER OF
                                                   ;  VECTORS) THAN THE OTHER
           (T (cond ((vect-diff vector-ends-out
                                (reverse vector-ends-back) x y diff-vect)
                     T)                            ; VECTORS ARE THE SAME
                    (T 'F)))))                     ; VECTORS ARE DIFFERENT

; VECT-DIFF
; SUM OF THE SQUARES OF THE DISTANCES BETWEEN END POINTS OF THE OUT AND
;  BACK VECTORS, TAKEN BETWEEN 1ST OUT FROM BALANCE POINT AND LAST ONE BACK
;  TO BALANCE POINT ETC.
(defun vect-diff (ind-o ind-b x y diff-vect)
  (setf sum 0)
  (dotimes (i (min (length ind-o) (length ind-b)))
           (setf sum (+ (sq (- (nth (setf j-o (nth i ind-o)) x)
                               (nth (setf j-b (nth i ind-b)) x)))
                        (sq (- (nth j-o y) (nth j-b y)))
                        sum)))
  (cond ((> sum diff-vect) 'F)
        (T 'T)))

; SQ
; RETURNS THE SQUARE
(defun sq (x) (* x x))

; AVERAGE
; AVERAGE OVER A LIST
; LENGTH MUST BE <= 127
(defun average (list)
  (if (> (setf length (length list)) 0)
      (/ (apply #'+ list) length)
      0))

; PROSP-FILTER
; RETURNS A LIST OF THOSE ELEMENTS OF LIST WHOSE CORRESPONDING ELEMENTS
;  OF PROSP? ARE T
(defun prosp-filter (structure list)
  (let ((prosp? (get-value structure prosp?))
        (prosp-list nil))
    (dotimes (i (length list))
      (when (nth i prosp?)
            (setf prosp-list (cons (nth i list) prosp-list))))
    (reverse prosp-list)))
```

```
; NOSP-FILTER
; RETURNS A LIST OF THOSE ELEMENTS OF LIST WHOSE CORRESPONDING ELEMENTS
;  OF NOSP? ARE T
(defun nosp-filter (structure list)
  (let ((nosp? (get-value structure nosp?))
        (nosp-list nil))
    (dotimes (i (length list))
      (when (nth i nosp?)
            (setf nosp-list (cons (nth i list) nosp-list))))
    (reverse nosp-list)))
```

```
; THE PROCEDURES IN THIS FILE FILL THE SLOTS IN NON-NOSP

; NON-NOSP-FILL
; THE BASIS FUNCTION CALCULATIONS ARE REDONE (WITH A SHORTER BASIS
;  FUNCTION LENGTH) FOR THE NON-NOSP EVENTS. THIS GIVE MORE RESOLUTION.
; THE NON-NOSP EVENTS ARE TESTED TO SEE IF THEY CURVE TOWARDS -X,
;  THEN THE NON-NOSP EVENTS ARE PROCESSED USING THE FAULT RULES
;  (ALL BUT THE FIRST TWO IN RULE-LIST)
(defun non-nosp-fill (name)
  (let* ((structure (eval name))
         (length (length (get-value structure tip-index))))
    (fill-non-nosp-slots structure length)      ; RE-DO THE EXPANSION IN TERMS
                                                ;  OF BASIS FUNCTIONS FOR
                                                ;  NON-NOSP EVENTS. PROCEDURES
                                                ;   FROM CALCPROP & NOSP
    (fill-curve-to--x structure length)         ; THIS IS NEEDED FOR THE WALL
                                                ; LOSS RULES (OD-ID ETC)
    (setf event-analysis (get-value structure list))
    (dotimes (i length)
      (dotimes (j (- (length rule-list) 2))     ; SEE EXPTLIST.LSP FOR A LIST
                                                ;  OF RULES.  THE FIRST TWO
                                                ;  ARE NOSP-ID AND PROSP-ID
                                                ;  WHICH ARE USED IN NOSP.LSP
      (setf rule (nth (+ 2 j) rule-list))
      (when (setf analysis-i                    ; EACH EVENT ALL RULES
                  (processor name rule i))
            (setf (nth i event-analysis)        ; RECORD ALL FAULTS THAT APPLY
                  (cons analysis-i
                        (nth i event-analysis))))))
   (put-value structure list (mapcar #'(lambda (x)
                                         (if (listp x)
                                             (reverse x)
                                             x))
                                     event-analysis))))


; FILL-NON-NOSP-SLOTS
; USED TO RECALCULATE THE BASIS VECTORS FOR NON-NOSP EVENTS TO GET BETTER
;  RESOLUTION
; IF THERE ARE NO NOSP EVENTS (IE A CALIBRATION TUBE) THEN THE
;  CORRESPONDING NOSP VALUES ARE COPIED
(defun fill-non-nosp-slots (structure length)
  (let* ((x-list (get-value structure x))
         (y-list (get-value structure y))
         (length-comp (get-value structure length-comp))
         (diff-vect (expt (* (get-value structure diff-vect)
                             (get-value structure min-x))
                          2)))
```

```
(setf nn-ve-out (make-list length                    ; SHORT FOR NON-NOSP VECTOR
                                                     ;  ENDS OUT & SIM. FOR REST
                           :initial-element nil)
      nn-ve-back (make-list length :initial-element nil)
      nn-b-out (make-list length :initial-element nil)
      nn-b-back (make-list length :initial-element nil)
      nn-bl-out (make-list length :initial-element nil)
      nn-bl-back (make-list length :initial-element nil)
      nn-ia-out (make-list length :initial-element nil)
      nn-ia-back (make-list length :initial-element nil)
      nn-o=b? (make-list length :initial-element nil))
(setf any-nosps? (if (> (length (get-value structure nosp-events)) 0)
                     T
                     nil))
(if
  any-nosps?                                         ; ARE THERE ANY NOSP EVENTS?
  (progn                                             ; YES - NOSP EVENTS => EXPAND
                                                     ; FOLLOWS NOSP-FILL
    (dotimes (i length)
      (when (not (nth i (get-value structure nosp?)))
            (setf x (nth i x-list))
            (setf y (nth i y-list))
            (setf tip-index (nth i (get-value structure tip-index)))
            (setf basis-length                       ; SAME AS IN CALCPROP.LSP
                  (expt (round
                          (* (get-value structure min-x)
                             (get-value structure basis-length)))
                        2))

            (setf basis-length                       ; DECREASE BASIS-LENGTH
                  (max 1 (round (/ basis-length (get-value structure
                                                  non-n-basis-length)))))
            (setf (nth i nn-ve-out) (find-string x y basis-length
                                                 tip-index))
            (setf (nth i nn-ve-back)
                  (complement (length x)
                              (find-string (reverse x)
                                           (reverse y)
                                           basis-length
                                           (- (length x) 1
                                              tip-index))))
            (setf (nth i nn-b-out)
                  (vect (get-value structure angles)
                        (calc-angles x y (nth i nn-ve-out))))
            (setf (nth i nn-b-back)
                  (vect (get-value structure angles)
                        (calc-angles x y (nth i nn-ve-back))))
            (setf (nth i nn-bl-out) (length (nth i nn-ve-out)))
```

```
                (setf (nth i nn-bl-back) (length (nth i nn-ve-back)))
                (setf (nth i nn-ia-out)
                      (to-degrees (get-value structure angles)
                                  (car (nth i nn-b-out))))
                (setf (nth i nn-ia-back)
                      (to-degrees (get-value structure angles)
                                  (car (nth i nn-b-back))))
                (setf (nth i nn-o=b?)
                      (out=back? x y (nth i nn-ve-out) (nth i nn-ve-back)
                                 length-comp diff-vect))))
          (put-value structure nn-ve-out nn-ve-out)
          (put-value structure nn-ve-back nn-ve-back)
          (put-value structure nn-b-out nn-b-out)
          (put-value structure nn-b-back nn-b-back)
          (put-value structure nn-bl-out nn-bl-out)
          (put-value structure nn-bl-back nn-bl-back)
          (put-value structure nn-ia-out nn-ia-out)
          (put-value structure nn-ia-back nn-ia-back)
          (put-value structure nn-o=b? nn-o=b?))
        (progn                                          ; NO NOSP EVENTS =>
                                                        ; COPY OLD VALUES
          (put-value structure nn-ve-out
                               (get-value structure vector-ends-out))
          (put-value structure nn-ve-back
                               (get-value structure vector-ends-back))
          (put-value structure nn-b-out
                               (get-value structure basis-out))
          (put-value structure nn-b-back
                               (get-value structure basis-back))
          (put-value structure nn-bl-out
                               (get-value structure basis-length-out))
          (put-value structure nn-bl-back
                               (get-value structure basis-length-back))
          (put-value structure nn-ia-out
                               (get-value structure initial-angle-out))
          (put-value structure nn-ia-back
                               (get-value structure initial-angle-back))
          (put-value structure nn-o=b?
                               (get-value structure out=back?))))))


; FILL-CURVE-TO--X
; PUTS VALUES IN THE NON-NOSP CURVE-TO--X-OUT & ..-BACK SLOTS
(defun fill-curve-to--x (structure length)
  (setf --x-out (make-list length :initial-element nil)
        --x-back (make-list length :initial-element nil))
  (setf ve-out (get-value structure nn-ve-out))
  (dotimes (i length)
```

```
    (when (nth i ve-out)                                ; IE WHEN NON-NIL THE EVENT
                                                        ;  IS A NON-NOSP
          (when (basis-test structure (nth i (get-value structure nn-b-out)))
                (setf (nth i --x-out) T))               ; TEST FOR CURVE TO -X
          (when (basis-test structure (nth i (get-value structure nn-b-back)))
                (setf (nth i --x-back) T))))
  (put-value structure wall-loss                        ; VALUE OF WALL LOSS PUT IN
                                                        ;  BY THE RULES
             (make-list length :initial-element nil))
  (put-value structure curve-to--x-out? --x-out)        ; RECORD
  (put-value structure curve-to--x-back? --x-back))

; BASIS-TEST
; TESTS TO SEE IF A TRACE CURVES TOWARD THE -X AXIS (IE CCW)
; ALGORITHM: AVERAGE ANGLE OF THE FIRST HALF OF THE BASIS VECTORS >
;            AVERAGE ANGLE OF THE LAST HALF OF THE BASIS VECTORS
;  (NOTE THAT THE ANGLES ARE WRT -X AXIS)
(defun basis-test (structure basis)
  (if (> (length basis) 1)
      (let ((angs (trans-to--x structure basis)))
        (if (> (avg-ang (butlast
                         angs (setf one-h (round (* .5 (length angs))))))
               (avg-ang (nthcdr (- (length angs) one-h) angs)))
               T
               nil))
      nil))

; AVG-ANG
; AVERAGE OVER A LIST
; LENGTH OF THE LIST <= 127
(defun avg-ang (ang-list)
  (if (> (length ang-list) 0)
      (/ (apply #'+ ang-list) (length ang-list))
      0))

; TRANS-TO--X
; MOVE BRANCH POINT FROM 0 TO -PI/2 (RADIANS)
(defun trans-to--x (structure angles)
  (let ((mxz (/ (setf angle (get-value structure angles)) 2))
        (nxz (/ (get-value structure angles) 4)))
    (mapcar #'(lambda (y) (if (and (>= y (- mxz))
                                   (< y (- nxz))) (+ y angle) y))
            (mapcar #'(lambda (x) (- mxz x)) angles))))
```

```
; ANALYSIS-FILL
(defun analysis-fill (name))

                                ; THIS SLOT IS FILLED BY OTHER
                                ;  PROCEDURES (STARTING WITH
                                ;  NOSP)
```

```
; THIS FILE CONATINS ALL THE PROCEDURES FOR DISPLAYING THE PLOTS OF THE
;  EVENTS, THE ANALYSIS AND PRINTING THE PLOTS, ANALYSIS AND STRUCTURE

; DISPLAY-RESULTS
(defun display-results (name &optional (plot-results? nil))
  (let ((structure (eval name)))
    (initgraphics 1)                                ; HI-RES SCREEN (HALO)
    (clk-wind 0)                                    ; WINDOW FOR DATE AND TIME
    (setf printer-toggle nil)                       ; PRINT EVENT INFO TO PRINTER
    (setf size-toggle nil)  ; SIZE FOR HALO SCREEN DUMP
    (let* ((per-page 12)  ; NUMBER OF EVENTS PLOTTED
           (pages (multiple-value-list              ;  SCREEN
                    (truncate (length (get-value structure tip-index))
                              per-page)))
           (number-of-pages (+ 1 (car pages))))
      (when (= (cadr pages) 0)                      ; 0 LEFT OVER
            (setf number-of-pages (car pages)))
      (dotimes (i (car pages))                      ; DO OVER THE # OF SCREENS
        (plot-screen name structure printer-toggle
                     (* i per-page) per-page number-of-pages (+ 1 i)
                     plot-results?))
      (when (> (cadr pages) 0)                      ; LAST PAGE
            (plot-screen name structure printer-toggle
                     (* (car pages) per-page) (cadr pages)
                    number-of-pages number-of-pages plot-results?))))
  (alpha))


; PLOT-SCREEN
; PLOTS A SINGLE SCREEN OF EVENTS
(defun plot-screen (name structure printer-toggle
                    start length number-of-pages page plot-results?)
  (setf screen-window (make-window-stream))
  (send screen-window :clear-screen)
  (plot-info name number-of-pages page)
  (setf x-start 150 x-inc 150 x-size 120        ; X-S../Y-S.. =2 FOR SQ ON TJET
        y-start 30 y-inc 50 y-size 60)          ; GRID FOR EVENT PLOTS
                                                ; .-START IS 0,0 FOR EVENT IN
                                                ;  UPPER LEFT
                                                ; .-INC IS SPACE BETWEEN EVENTS
                                                ; .-SIZE IS SIZE OF 1 EVENT
  (setf min-x (get-value structure min-x))      ; FOR SCALING
  (if (= (length (get-value structure           ; FOR CALIBRATION TUBES USE A
                            nosp-events)) 0)    ;  DIFFERENT SCALING
      (setf size (max (abs (get-value structure max-x))
                      (abs (get-value structure min-x))
                      (abs (get-value structure max-y))
                      (abs (get-value structure min-y))))
```

```
    (setf size (abs min-x)))                          ; TUBES WITH NOSPS
 (setf sx (/ x-size size)                             ; SCALE FOR SCREEN PLOTS
       sy (/ y-size size))
 (setf x (get-value structure x)
       y (get-value structure y))
 (dotimes (j length)                                  ; DO OVER # OF EVENTS ON PAGE
   (let* ((i (+ start j))                             ; EVENT #
          (qr (multiple-value-list (truncate j 4)))
          (cx (+ x-start (* (cadr qr) x-inc))); ZERO POINT FOR THE EVENT
          (cy (+ y-start (* (car qr) y-inc)))
          (x-data (inside 639                         ; TRANSLATE TO SCREEN CO-ORDS
                    (screen-co-ords '+ cx sx (nth i x))))
          (y-data (inside 198
                    (screen-co-ords '- cy sy (nth i y)))))
     (plot-cross 4 cx cy)                             ; CROSS AT EVENT ORIGIN
     (print-number i cx cy (nth i (get-value structure list)))
     (dotimes (k (- (length x-data) 1))               ; PLOT EVENT ON SCREEN
       (plot-line k x-data y-data))))
(if plot-results?                                     ; T => DUMP SCREEN TO PRINTER
                                                      ;  WITH NO WAITING
    (progn (setpattr '(0 0 0 2 2 0 1 0))              ; DOUBLE SIZE
           (timer)                                    ; PUT TIME & DATE ON SCREEN
           (gprint)                                   ; DUMP TO PRINTER (HALO)
           (setf stream (open (pathname "lpt1") :direction ':output))
           (when (or (= page number-of-pages) ; LAST SCREEN FULL
                     (evenp page))                    ; 2 DUMPS PER PAGE
                     (princ (string 12)               ; FORM FEED
    stream))
           (close stream))
    (catch 'up (loop                                  ; NIL => ANALYST CONTROLS
                 (cond ((not (setf return  ; LISTEN-TIME-CLEAR-PRINT WAITS
                                                      ;  FOR A KEY TO BE PRESSED AND
                                                      ;  RETURNS A STRING CONTAINING
                                                      ;  THE KEY
                             (listen-time-clear-print)))
                        (throw 'up nil))
                       ((equal return "p"); LISTEN... RETURNS "p"
                                                      ; TOGGLE THE PRINTER TOGGLE
                                                      ;  BETWEEN ON & OFF (SEE BELOW)
                         (setf printer-toggle
                               (toggle printer-toggle)))
                       ((equal return "a"); PRINT ALL THE STRUCTURED INFO
                                                      ;  ON THE TUBE
                        (print-averages name structure)
                        (dotimes (i (length
                                      (get-value structure tip-angle)))
                          (print-features name i structure)))
                       ((equal return "g"); SCREEN DUMP TO PRINTER
```

```
 (gprint))
((equal return "s"); TOGGLE THE SIZE OF THE SCREEN
                   ;  DUMP BETWEEN 1 & 4 DOTS PER
                   ;  PIXEL
 (setf size-toggle (toggle size-toggle))
 (if size-toggle
     (setpattr '(0 0 0 2 2 0 1 0))
     (setpattr '(0 0 0 1 1 0 1 0))))
((equal return "t"); PLOT AN X AT THE TIP OF
                   ;  EACH EVENT
 (dotimes (j length)
   (let* ((i (+ start j))
     (qr (multiple-value-list (truncate j 4)))
     (cx (+ x-start (* (cadr qr) x-inc)))
     (cy (+ y-start (* (car qr) y-inc)))
     (x-data (inside 639
               (screen-co-ords '+ cx sx (nth i x))))
     (y-data (inside 198
              (screen-co-ords '- cy sy (nth i y)))))
   (plot-x 4 (nth (nth i
        (get-value structure tip-index)) x-data)
             (nth (nth i
        (get-value structure tip-index))
        y-data)))))
((or (< (setf num (read-from-string return))
        (+ 1 start)) ; NEXT SCREEN: NUM NOT SHOWN
        (> num (+ start length))) (throw 'up nil))
(T                   ; NUM ON SCREEN
                     ; IF PRINTER-TOGGLE IS T:
                     ;  PRINT STRUCTURED INFO ON
                     ;  EVENT NUM
  (when printer-toggle (print-features name
                           (- num 1) structure))
                     ; PLOT AT BOTTOM OF SCREEN
    (plot-details x-start 185 x-inc y-inc sx sy
        (- num 1) structure)))))))

; PRINT-FEATURES
; PRINTS SELECTED VALUES FROM THE TUBE STRUCTURE FOR EVENT I
;  - SEE OPTION A IN PRINT-SCREEM
(defun print-features (name i structure)
  (setf stream (open (pathname "lpt1") :direction ':output))
  (terpri stream)
  (princ "FILE NAME  " stream) (princ name stream)
  (terpri stream)
  (princ "EVENT  " stream) (princ              ; FOR DISPLAY ONLY
                               ; INTERNALLY START WITH 0
                          (+ 1 i) stream)
```

```
(terpri stream)
(princ "TIP-INDEX  " stream) (princ (nth i (get-value structure tip-index))
                                    stream)
(terpri stream)
(princ "TIP-ANGLE  " stream) (princ (nth i (get-value structure tip-angle))
                                    stream)
(terpri stream)
(princ "TIP-LENGTH  " stream)
(princ (nth i (get-value structure tip-length)) stream)
(terpri stream)
(princ "VECTOR-ENDS-OUT  " stream)
(princ (nth i (get-value structure vector-ends-out)) stream)
(terpri stream)
(princ "VECTOR-ENDS-BACK  " stream)
(princ (nth i (get-value structure vector-ends-back)) stream)
(terpri stream)
(princ "BASIS-OUT    " stream)
(princ (setf basis+ (nth i (get-value structure basis-out))) stream)
(terpri stream)
(princ "             " stream)
(princ (trans-+-- basis+ (get-value structure angles)) stream)
(terpri stream)
(princ "BASIS-BACK   " stream)
(princ (setf basis+ (nth i (get-value structure basis-back))) stream)
(terpri stream)
(princ "             " stream)
(princ (trans-+-- basis+ (get-value structure angles)) stream)
(terpri stream)
(princ "BASIS-LENGTH-OUT  " stream)
(princ (nth i (get-value structure basis-length-out)) stream)
(terpri stream)
(princ "BASIS-LENGTH-BACK " stream)
(princ (nth i (get-value structure basis-length-back)) stream)
(terpri stream)
(princ "INITIAL-ANGLE-OUT  " stream)
(princ (nth i (get-value structure initial-angle-out)) stream)
(terpri stream)
(princ "INITIAL-ANGLE-BACK " stream)
(princ (nth i (get-value structure initial-angle-back)) stream)
(terpri stream)
(princ "S-SHAPE-OUT?  " stream)
(princ (nth i (get-value structure s-shape-out?)) stream)
(terpri stream)
(princ "S-SHAPE-BACK? " stream)
(princ (nth i (get-value structure s-shape-back?)) stream)
(terpri stream)
(princ "OUT=BACK?  " stream)
(princ (nth i (get-value structure out=back?)) stream)
```

```
(terpri stream)
(princ "PROSP?  " stream) (princ (nth i (get-value structure prosp?))
                                 stream)
(terpri stream)
(princ "NOSP?   " stream) (princ (nth i (get-value structure nosp?))
                                 stream)
(terpri stream)
(when (not (nth i (get-value structure nosp?)))
  (progn  (princ "NN-VE-OUT  " stream)
          (princ (nth i (get-value structure nn-ve-out)) stream)
          (terpri stream)
          (princ "NN-VE-BACK  " stream)
          (princ (nth i (get-value structure nn-ve-back)) stream)
          (terpri stream)
          (princ "NN-B-OUT  " stream)
          (princ (nth i (get-value structure nn-b-out)) stream)
          (terpri stream)
          (princ "NN-B-BACK  " stream)
          (princ (nth i (get-value structure nn-b-back)) stream)
          (terpri stream)
          (princ "NN-BL-OUT  " stream)
          (princ (nth i (get-value structure nn-bl-out)) stream)
          (terpri stream)
          (princ "NN-BL-BACK  " stream)
          (princ (nth i (get-value structure nn-bl-back)) stream)
          (terpri stream)
          (princ "NN-IA-OUT  " stream)
          (princ (nth i (get-value structure nn-ia-out)) stream)
          (terpri stream)
          (princ "NN-IA-BACK  " stream)
          (princ (nth i (get-value structure nn-ia-back)) stream)
          (terpri stream)
          (princ "NN-O=B?  " stream)
          (princ (nth i (get-value structure nn-o=b?)) stream)
          (terpri stream)
          (princ "CURVE-TO--X-OUT?  " stream)
          (princ (nth i (get-value structure curve-to--x-out?)) stream)
          (terpri stream)
          (princ "CURVE-TO--X-BACK?  " stream)
          (princ (nth i (get-value structure curve-to--x-back?)) stream)
          (terpri stream)
          (princ "WALL-LOSS  " stream)
          (princ (nth i (get-value structure wall-loss)) stream)
          (terpri stream)))
(princ "ANALYSIS   " stream)
(princ (nth i (get-value structure list)) stream)
(terpri stream)  (terpri stream)  (terpri stream)  (terpri stream)
(ignore-errors (close stream)))
```

```
; PRINT-AVERAGES
; PRINTS INFO FROM INFO STRUCTURE FOR EVENT I
(defun print-averages (name structure)
  (setf stream (open (pathname "lpt1") :direction ':output))
  (terpri stream)
  (princ "FILE NAME  " stream) (princ name stream)
  (terpri stream)
  (princ "TUBE NUMBER  " stream)
  (princ (get-value structure tube-number) stream) (terpri stream)
  (princ "STATION  " stream)
  (princ (get-value structure station) stream) (terpri stream)
  (princ "UNIT  " stream)
  (princ (get-value structure unit) stream) (terpri stream)
  (princ "HEAT EXCHANGER TYPE  " stream)
  (princ (get-value structure heat-exchanger-type) stream) (terpri stream)
  (princ "HEAT EXCHANGER NUMBER  " stream)
  (princ (get-value structure heat-exchanger-number) stream) (terpri stream)
  (princ "FREQUENCY  " stream)
  (princ (get-value structure frequency) stream) (terpri stream)
  (princ "DATE  " stream)
  (princ (get-value structure date) stream) (terpri stream)
  (princ "TIME  " stream)
  (princ (get-value structure time) stream) (terpri stream)
  (princ "SUPERVISOR  " stream)
  (princ (get-value structure supervisor) stream) (terpri stream)
  (princ "COMMENT  " stream)
  (princ (get-value structure comment) stream) (terpri stream)
  (princ "AVERAGE TIP LENGTH  " stream)
  (princ (get-value structure nosp-avg-tip-length) stream)
  (terpri stream)
  (princ "AVERAGE TIP ANGLE  " stream)
  (princ (get-value structure nosp-avg-tip-angle) stream)
  (terpri stream)
  (princ "NORMAL SUPPORT PLATES  " stream)
  (princ (mapcar #'(lambda (x) (1+ x)) (get-value structure nosp-events))
         stream)
  (terpri stream)
  (ignore-errors (close stream)))


; TOGGLE
; TOGGLES TOGGLE BETWEEN T & NIL
(defun toggle (toggle)
  (cond (toggle nil)
        (T T)))
```

```
; PLOT-DETAILS
; AT THE BOTTOM OF THE SCREEN, PLOTS THE OUT & BACK BASIS VECTORS TOGETHER
;  & SEPARATELY
(defun plot-details (x-start y-start x-inc y-inc sx sy i structure)
  (%draw-line 50 155 550 155 1 0)
  (setf info-window (make-window-stream
         :top  20
         :left 0
         :height 5
         :width 80))
  (send info-window :clear-screen)
  (send info-window :set-cursorpos 30 0)
  (princ "DETAILS FOR EVENT  " info-window)
  (princ (+ 1 i) info-window)
  (send info-window :set-cursorpos (setf xs (- (round (/ x-start 8)) 8)) 5)
  (princ "OUT" info-window)
  (send info-window :set-cursorpos (+ xs (setf xi (round (/ x-inc 8)))) 5)
  (princ "BACK" info-window)
  (send info-window :set-cursorpos (+ xs (* 2 xi)) 5)
  (princ "BOTH" info-window)
  (when (and (> (car (setf out (nth i (get-value structure vector-ends-out))))
                0)                                    ; FOR PLOT-DET
             (> (car (setf back (nth i
                                     (get-value structure vector-ends-back))))
                0))
        (plot-det x-start y-start x-inc
                  sx sy out i structure)
        (plot-det (+ x-inc x-start) y-start x-inc
                  sx sy back i structure)
        (plot-det (+ (* 2 x-inc) x-start) y-start x-inc
                  sx sy out i structure)
        (plot-det (+ (* 2 x-inc) x-start) y-start x-inc
                  sx sy back i structure)
       (plot-det (+ (* 2 x-inc) x-start) y-start x-inc
                 sx sy
                 (list (nth i (get-value structure tip-index))) i structure)))

; PLOT-DET
; DOES THE PLOTS OF THE BASIS VECTORS FOR PLOT-DETAILS
(defun plot-det (x-start y-start x-inc sx sy filter i structure)
    (let* ((cx x-start)
           (cy y-start)
           (x-data (inside 639                        ; MAKE SURE ON SCREEN
                   (screen-co-ords                    ; TRANS TO SCREEN CO-ORDS
                     '+ cx sx                         ; ZERO,SCALE
                     (add-zero                        ; FOR 1ST BASIS VECTOR
                        (filter-points                ; SEPARATE OUT BASIS VECTORS
                         filter                       ; VECTOR ENDS
```

```
                        (nth i (get-value structure x)))))))
          (y-data (inside 199
                  (screen-co-ords
                    '- cy sy
                    (add-zero
                      (filter-points
                       filter
                       (nth i (get-value structure y)))))))
        (plot-cross 4 cx cy)
        (dotimes (k (- (length x-data) 1))
          (plot-line k x-data y-data))))

; ADD-ZERO
; ADDS A ZERO TO THE FRONT OF LIST - SEE PLOT-DET
(defun add-zero (list)
  (cons 0 list))

; FILTER-POINTS
; RETURNS THOSE VALUES OF POINTS WHOSE INDICES ARE IN INDEX-LIST
(defun filter-points (index-list points)
  (setf filtered-list nil)
  (dotimes (i (length index-list))
           (setf filtered-list (cons (nth (nth i index-list) points)
                                     filtered-list)))
  (reverse filtered-list))


; PLOT-INFO
; PLOTS THE HEADER ACROSS THE TOP OF THE SCREEN
(defun plot-info (name number-of-pages page)
  (setf info-window (make-window-stream
        :top 0
        :width 50
        :left 5
        :height 1))
  (send info-window :set-cursorpos 0 0)
  (princ (string name) info-window)
  (send info-window :set-cursorpos 25 0)
  (princ (string-append (string-num page) " of "
                        (string-num number-of-pages))
         info-window))

; PLOT-CROSS
; PLOTS A CROSS AT CX,CY OF SIZE N
(defun plot-cross (n cx cy)
  (let ((nx (round (* 2.5 n))) (ny n))
    (%draw-line cx (+ cy ny) cx (- cy ny) 1 0)
    (%draw-line (- cx nx) cy (+ cx nx) cy 1 0)))
```

```
; PLOT-X
; PLOTS AN X AT CX,CY OF SIZE N
(defun plot-x (n cx cy)
  (let ((nx (round (* 2 n))) (nxh n) (ny n) (nyh (round (* .5 n))))
    (%draw-line (+ cx nx) (+ cy ny) (+ cx nxh) (+ cy nyh) 1 0)
    (%draw-line (- cx nx) (- cy ny) (- cx nxh) (- cy nyh) 1 0)
    (%draw-line (- cx nx) (+ cy ny) (- cx nxh) (+ cy nyh) 1 0)
    (%draw-line (+ cx nx) (- cy ny) (+ cx nxh) (- cy nyh) 1 0)))

; PLOT-LINE
; PLOTS A LINE BETWEEN THE ITH AND I+1TH VALUES OF X,Y
(defun plot-line (i x y)
  (%draw-line (nth i x) (nth i y) (nth (1+ i) x) (nth (1+ i) y) 1 0))

; SCREEN-CO-ORDS
; TRANSLATES THE ELEMENTS OF LIST TO SCREEN CO-ORDINATES
(defun screen-co-ords (sign c s list)
  (mapcar #'(lambda (x) (round (funcall sign c (* x s)))) list))

; INSIDE
; MAKES SURE ALL THE ELEMENTS OF LIST ARE BETWEEN 0 AND END
; CALLED BY PLOT-DET
(defun inside (end list)
  (mapcar #'(lambda (x) (cond ((< x 0) 0)
                              ((> x end) end)
                              (T x))) list))

; PRINT-NUMBER
; PRINTS THE EVENT NUMBER AND ANALYSIS BELOW EVENT ON THE SCREEN
; PLOTS A BOX FOR NON SP AND NON HALF SP EVENTS
; IF AN EVENT HAS MORE THAN ONE INTERPRETATION THE FIRST ONE IS PRINTED
(defun print-number (i cx cy analysis)
  (setf event-window (make-window-stream
        :top  (+ 1 (round (/ cy 8))) ; 8 POINTS PER CHARACTER
        :left (- (round (/ cx 8)) 11)
        :height 1
        :width 16
        :attribute 64))
  (when (equal analysis nil)
        (setf analysis "???"))
  (when (listp analysis)
        (setf analysis (car analysis)))
  (when (not (or (equal analysis "half sp")
                 (equal analysis "sp")))
        (box-around event-window))
  (send event-window :set-cursorpos 0 0)
  (setf event-string (string-append (string-num (+ 1 i)) " "
```

```
                                        analysis))

  (princ (string-append (make-array (max 1 (round
                                             (* .5 (- 16 (length event-string)))))
                                    :element-type 'string-char
                                    :initial-element 32)
                        event-string) event-window))

; BOX-AROUND
; PLOTS A BOX AROUND THE WINDOW WINDOW
(defun box-around (window)
  (let* ((size (multiple-value-list (send window :size)))
         (position (multiple-value-list (send window :position)))
         (x-zer (- (* 8 (- (car position) 1)) 1))
         (y-zer (- (* 8 (cadr position)) 3))
         (x-side (+ 2 (* 8 (car size))))
         (y-side (+ 4 (* 8 (cadr size)))))
    (%draw-line x-zer y-zer (+ x-zer x-side) y-zer 1 0)
    (%draw-line (+ x-zer x-side) y-zer (+ x-zer x-side) (+ y-zer y-side) 1 0)
    (%draw-line (+ x-zer x-side) (+ y-zer y-side) x-zer (+ y-zer y-side) 1 0)
    (%draw-line x-zer (+ y-zer y-side) x-zer y-zer 1 0)))

; STRING-NUM
; RETURNS THE NUMBER N AS A STRING
(defun string-num (n)
  (cond ((< n 0) (string-append                          ; DOESN'T WORK
                              "-" (string-num (- n))))
        ((= n 0) "0")
        ((and (> n 0) (< n 10)) (string (+ 48 n)))
        (T (let ((x (truncate (log n 10))))
             (let ((y (multiple-value-list (truncate n (expt 10 x)))))
               (string-append (+ 48 (car y)) (string-num (cadr y))))))))

; TIMER
; PRINTS THE TIME AND DATE TO THE UPPER RIGHT CORNER OF THE SCREEN
(defun timer ()
    (send clock-window :set-cursorpos 0 0)
    (setf time-list (multiple-value-list (get-decoded-time)))
    (setf t1 (nth 2 time-list))
    (cond ((> t1 11) (setq ampm 'pm) (setq t1 (- t1 12)))
          ((< t1 12) (setq ampm 'am)))
    (setq colon (make-symbol ":"))
    (setq tim (list t1 (nth 1 time-list) (nth 0 time-list) ampm))
    (send clock-window :set-cursorpos 0 0)
    (cond ((> t1 9) (princ (car tim) clock-window))
          (t (princ (make-symbol (string-append " "
               (string (+ (car tim) 48)))) clock-window)))
    (send clock-window :set-cursorpos 2 0)
```

```
      (princ colon clock-window)
      (send clock-window :set-cursorpos 3 0)
      (cond ((> (cadr tim) 9) (princ (cadr tim) clock-window))
            (t (princ (make-symbol (string-append "0"
                 (string (+ (cadr tim) 48)))) clock-window)))
      (send clock-window :set-cursorpos 5 0)
      (princ colon clock-window)
      (send clock-window :set-cursorpos 6 0)
      (cond ((> (caddr tim) 9) (princ (caddr tim) clock-window))
            (t (princ (make-symbol (string-append "0"
                 (string (+ (caddr tim) 48)))) clock-window)))
      (send clock-window :set-cursorpos 8 0)
      (princ " " clock-window)
      (send clock-window :set-cursorpos 9 0)
      (princ (nth 3 tim) clock-window))

; DATER
; RETURNS THE DATE AS A STRING
(defun dater ()
  (setf time-list (multiple-value-list (get-decoded-time)))
  (setf month '(JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC))
  (setf date (string-append (string-num (nth 3 time-list))
                            " "
                            (string (nth (- (nth 4 time-list) 1) month))
                            " "
                            (string-num (nth 5 time-list))))
  (send date-window :set-cursorpos 0 0)
  (princ date date-window))

; LISTEN-TIME-CLEAR-PRINT
; PRINTS THE TIME AND DATE IN THE UPPER RIGHT CORNER AND WAIT FOR A KEY
;  TO BE PRESSED.  IF THE KEY IS p, g, a, s or t THEN THAT LETTER IS RETURNED
;  AS A STRING.  IF A DIGIT THEN THE PROGRAM WAITS FOR A SECOND KEY TO BE
;  PRESSED.  IF THE SECOND KEY IS ENTER THEN THE DIGIT IS RETURNED AS A
;  STRING.  IF THE SECOND KEY IS A DIGIT THEN A STRING IS RETURNED CONTAINING
;  THE 2 DIGITS.  OTHERWISE NIL IS RETURNED.
(defun listen-time-clear-print ()
       (cursor-off)
       (dater)                                          ; PRINT DATE
       (catch 'up (loop (timer)                         ; PRINT TIME
              (dotimes (wait 50))                       ; WAIT FOR 50 UNITS
              (when (send *terminal-io* :listen)        ; CHECK KEYBOARD BUFFER FOR
                                                        ;  KEYSTROKE - IF NONE LOOPS
                                                        ;  TO TIMER AND PRINTS NEW
                                                        ;  TIME
                    (setf charl                         ; IF KEY - READ IT
                              (send *terminal-io* :read-char))
                    (when (not (test-no charl))  ; TEST KEY FOR DIGIT
```

```
                    (when (= char1 112) (setf return "p")
                                        (throw 'up nil))
                    (when (= char1 103) (setf return "g")
                                        (throw 'up nil))
                    (when (= char1 97) (setf return "a")
                                        (throw 'up nil))
                    (when (= char1 115) (setf return "s")
                                        (throw 'up nil))
                    (when (= char1 116) (setf return "t")
                                        (throw 'up nil))
                    (setf return nil) (throw 'up nil))
                (setf char2                    ; READ SECOND KEY
                    (send *terminal-io* :read-char))
                (cond ((= char2 10)            ; CHECK FOR ENTER
                                (setf return (string char1))
                                (throw 'up nil))
                    ((test-no char2)           ; CHECK FOR DIGIT
                     (setf return (string-append
                                    (string char1)
                                    (string char2)))
                                (throw 'up nil))
                    (T (setf return nil) (throw 'up nil))))))
      (cursor-on)
      (send clock-window :clear-input)
  return)

; TEST-NO
; CHECK CHAR TO SEE IF IT IS A DIGIT
(defun test-no (char)
  (and (>= char 48) (<= char 57)))          ; STRING CODE FOR NUMBERS

; CLK-WIND
; WINDOW FOR DATE AND TIME INFO
; PAGE IS THE TEXT SCREEN PAGE NUMBER
(defun clk-wind (page)
 (setf clock-window (make-window-stream
      :cursorpos-x   0
      :cursorpos-y   0
      :attribute     7
      :left          65
      :top           0
      :width         13
      :status        4
      :height        1
      :page          page))
  (setf date-window (make-window-stream
      :left 50
      :top 0
```

```
:width 12
:height 1
:page page)))
```

```
; RULE
; STRUCTURE TO STORE RULES FOR ANALYZING EVENTS
; THE ONLY ARGUMENT FOR THE PREDICATES AND THE THEN PROCEDURE IS A TUBE
;   STRUCTURE
; IF ALL THE IF PREDICATES ARE TRUE THE THEN PROCEDURE IS CALLED
; THE IFS-ENGLISH-TRUE AND IFS-ENGLISH-FALSE ARE RECORDED FOR EACH EVENT AS A
;   RECORD OF WHICH PREDICATES ARE TESTED
; NOTE THAT THE USE OF THE IFS-ENGLISH-FALSE ISN'T IMPLEMENTED YET
; THE IFS-ENGLISH-TRUE AND THEN-ENGLISH-TEMPLATE ARE USED TO CREATE A FILE
;   CONTAINING ALL THE ENGLISH VERSIONS OF THE RULES  LISTED IN RULE-LIST
; NOTE THAT IFS-ENGLISH-TRUE AND IFS-ENGLISH-FALSE ARE LISTS OF STRINGS SO
;   TO GET PROPER SPACING WHEN PRINTED IN A
;
; AND   .........
;       ......... ETC.
;
;   FORMAT, THE " SHOULD BE IN COLUMN 7 AND SUBSEQUENT LINES SHOULD HAVE
;   THE FIRST CHARACTER UNDER THE FIRST CHARACTER IN THE PERVIOUS LINE.  THUS
;   TO PRINT AS ABOVE APPEND "AND   " TO THE STRINGS
(defstruct (rule)
  (ifs nil)                                    ; LIST OF PREDICATES WITH ARGS
                                               ;  (STRUCTURE I) SEE
                                               ;  RULE-PROCESSOR
  (ifs-english-true nil)                       ; LIST (SAME LENGTH)
  (ifs-english-false nil)
  (then nil)                                   ; PROCEDURE THAT RETURNS
                                               ;   ANALYSIS - FOR DISPLAY
                                               ;   ARGS: (STRUCTURE I)
  (then-english-template nil))                 ; ANALYSIS WITH # FOR
                                               ;   VARIABLES - USED BY FILE
                                               ;   WITH ENGLISH VERSION OF
                                               ;   RULES


; RULE-PROCESSOR
; PROCESSES RULE RULE USING STRUCTURE STRUCTURE
; RETURNS ANALYSIS GIVEN BY THEN PROCEDURE IF ALL THE IFS PREDICATES ARE T
;  OTHERWISE RETURNS NIL
(defun processor (structure-name rule-name i)
  (let ((structure (eval structure-name))
        (rule (eval rule-name)))
    (if (our-apply                             ; APPLY DOESN'T WORK ON AND
                                               ;  SINCE ITS A SPECIAL FORM
                  #'and (mapcar #'(lambda (x) (funcall x structure i))
                                (rule-ifs rule)))
        (funcall (rule-then rule) structure i)
```

```
        nil)))

; OUR-APPLY
; APPLY FOR SPECIAL FORMS (SUCH AS AND)
(defun our-apply (function list)
  (eval (cons function list)))

; RULE-WRITER
; WRITES TO FILE RULE-ENG.LSP THE ENGLISH VERSION OF THE RULES IN
;   RULE-LIST
(defun rule-writer ()
  (setf stream (open (pathname "d:\\lsp-2\\eddy\\rule-eng.lsp")
                     :direction ':output))
  (terpri stream)
  (terpri stream)
  (terpri stream)
  (princ
"         NOTES:

           1. THE SYMBOLS IN LOWER CASE REFER TO CONSTANTS IN THE TUBE
              DATA STRUCTURE.

           2. THE NAME IN SQUARE BRACKETS AFTER THE RULE NUMBER IS THE NAME
              OF THE LISP SOURCE FILE WHERE THE RULE IS DEFINED AND
              DOCUMENTED.  THE PREDICATES (SEE 3.) WILL BE IN THIS FILE OR
              A REFERENCE WILL BE MADE TO THE FILE THAT CONTAINS THEM.

           3. THE NAME IN SQUARE BRACKETS AFTER EACH LINE IN THE IF LIST IS
              THE CORRESPONDING PREDICATE."
           stream)
  (terpri stream)
  (terpri stream)
  (dotimes (i (length rule-list))
    (terpri stream)
    (princ (string-append " RULE " (string-num (+ 1 i)) "     ["
                          (symbol-name (nth i rule-list)) ".LSP]") stream)
    (terpri stream)
    (terpri stream)
    (princ "  IF" stream)
    (terpri stream)
    (setf rule-i (eval (nth i rule-list)))
    (setf rule-english (rule-ifs-english-true rule-i))
    (setf rule-predicates (rule-ifs rule-i))
    (princ (string-append "       " (car rule-english)) stream)
    (terpri stream)
    (princ (string-append "       "
                          "[" (symbol-name (car rule-predicates)) "]") stream)
    (terpri stream)
```

```
    (dotimes (j (- (length (rule-ifs rule-i)) 1))
      (princ (string-append "   AND " (nth (1+ j) rule-english)) stream)
      (terpri stream)
      (princ (string-append "       "
                            "[" (symbol-name (nth (1+ j) rule-predicates))
                             "]") stream)
      (terpri stream))
    (terpri stream)
    (princ "  THEN" stream)
    (terpri stream)
    (princ (string-append "       "
                          (rule-then-english-template rule-i)) stream)
    (terpri stream)
    (princ (string-append "       "
                          "[" (symbol-name (rule-then rule-i)) "]") stream)
    (terpri stream)
    (terpri stream)
    (princ
"                                - - + - -" stream)
    (terpri stream)
    (terpri stream))
  (close stream)
  (sys:8087-fpp :use))
```

APPENDIX G

RULE FILES

```
; PROSP-ID
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf prosp-id
      (make-rule
        :ifs '(t-a20 ia-q o=b s-shape)
        :ifs-english-true '(
      "THE TIP ANGLE IS WITHIN + OR - 20 DEGREES OF THE -X AXIS"
      "THE INITIAL ANGLE (BOTH OUT AND BACK) IS IN THE 1ST OR 2ND QUADRANT"
      "THE OUT AND BACK SIGNALS ARE THE SAME"
      "THE S-SHAPE IS PRESENT IN BOTH THE OUT AND BACK SIGNALS")
        :ifs-english-false '(
      "THE TIP ANGLE IS NOT WITHIN + OR - 20 DEGREES OF THE -X AXIS"
      "THE INITIAL ANGLE (BOTH OUT AND BACK) IS NOT WITHIN THE 1ST OR 2ND
       QUADRANT"
      "THE OUT AND BACK SIGNALS ARE DIFFERENT"
      "THE S-SHAPE IS MISSING FROM BOTH THE OUT AND BACK SIGNALS")
        :then 'prosp-analysis
        :then-english-template
      "THE EVENT IS PROVISIONALLY DUE TO A NORMAL SUPPORT PLATE (PROSP)"))

;------------------------------ T-A20 ---------------------------------------

; T-A20
; PREDICATE THAT RETURNS T IF THE TIP-ANGLE IS WITHIN 20
;   DEGREES OF THE -X AXIS
(defun t-a20 (structure i)
  (cond ((between (nth i (get-value structure tip-angle)) -20 20) T)
        (T nil)))

; BETWEEN
; CALLED BY T-A20
(defun between (angle limit-1 limit-2)
  (cond ((and (>= angle limit-1)
              (<= angle limit-2))
         T)
        (T nil)))


;---------------------------------- IA-Q ------------------------------------

; IA-Q
; PREDICATE THAT RETURNS T IF INITIAL-ANGLE IS IN THE FIRST OR SECOND QUADRANT
;  FOR BOTH OUT AND BACK
(defun ia-q (structure i)
  (cond ((and (between (nth i
                            (get-value structure initial-angle-out)) 0 180)
              (between (nth i
```

```
                        (get-value structure initial-angle-back)) 0 180))
 T)
       (T nil)))

;------------------------------------ O=B ----------------------------------

; O=B
; PREDICATE THAT RETURNS T IF OUT=BACK? IS TRUE
; NOTE THAT OUT=BACK? HAS A VALUE OF T OR F
(defun o=b (structure i)
  (cond ((equal (nth i (get-value structure out=back?)) 'T) T)
        (T nil)))

;---------------------------------- S-SHAPE --------------------------------

; S-SHAPE
; PREDICATE THAT RETURNS T IF S-SHAPE-OUT? AND S-SHAPE-BACK? ARE BOTH TRUE
; NOTE THAT S-SHAPE-.. HAS A VALUE OF T OR F
(defun s-shape (structure i)
  (cond ((and (not (equal (nth i (get-value structure s-shape-out?)) 'F))
              (not (equal (nth i (get-value structure s-shape-back?)) 'F)))
         T)
        (T nil)))

; PROSP-ANALYSIS
; ONLY CALLED IF EVENT I PASSES THE RULE FOR PROSPS
(defun prosp-analysis (structure i)
  "PROSP")
```

```
; NOSP-ID
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf nosp-id
      (make-rule
        :ifs '(prosp-test prosp-avg-len prosp-avg-angle)
        :ifs-english-true '(
      "THE EVENT IS A PROSP"
      "THE TIP LENGTH IS WITHIN nosp-length (FRACTION) OF THE PROSP
       AVERAGE TIP LENGTH"
      "THE TIP ANGLE IS WITHIN nosp-angle DEGREES OF THE PROSP AVERAGE
       TIP ANGLE")
        :ifs-english-false '(
      "THE EVENT IS NOT A PROSP"
      "THE TIP LENGTH IS NOT WITHIN nosp-length (FRACTION) OF THE PROSP
       AVERAGE TIP
       LENGTH"
      "THE TIP ANGLE IS NOT WITHIN nosp-angle DEGREES OF THE PROSP AVERAGE
       TIP ANGLE")
        :then 'nosp-analysis
        :then-english-template
      "THE EVENT IS DUE TO A NORMAL SUPPORT PLATE (NOSP)"))


; PROSP-TEST
; PREDICATE THAT RETURNS T IF ITH EVENT IS A PROSP
(defun prosp-test (structure i)
  (nth i (get-value structure prosp?)))

; PROSP-AVG-LEN
; PREDICATE THAT RETURNS T IF TIP LENGTH IS WITHIN
;  NOSP-LENGTH OF THE PROSP AVERAGE TIP LENGTH
(defun prosp-avg-len (structure i)
   (let* ((prosp-avg-len (get-value structure prosp-avg-tip-length))
          (tol (* (get-value structure nosp-length) prosp-avg-len))
          (tip-len (nth i (get-value structure tip-length))))
     (if (and (>= tip-len (- prosp-avg-len tol))
              (<= tip-len (+ prosp-avg-len tol)))
         T
         nil)))

; PROSP-AVG-ANGLE
; PREDICATE THAT RETURNS T IF AVEARGE TIP ANGLE IS WITHIN
;  NOSP-ANGLE OF THE AVERAGE PROSP TIP ANGLE
(defun prosp-avg-angle (structure i)
   (let* ((prosp-avg-angle (get-value structure prosp-avg-tip-angle))
          (tol (get-value structure nosp-angle))
```

```
        (tip-angle (nth i (get-value structure tip-angle))))
    (if (and (>= tip-angle (- prosp-avg-angle tol))
             (<= tip-angle (+ prosp-avg-angle tol)))
        T
        nil)))

; NOSP-ANAYLSIS
; ONLY CALLED IF EVENT I PASSES THE RULE FOR NOSPS
(defun nosp-analysis (structure i)
  "sp")
```

```
; OD-ID
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf od-id
       (make-rule
         :ifs '(non-nosp curve-to--x? angle-to-tip-od? o=b?)
         :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THE OUT OR BACK SIGNAL CURVES TOWARDS -X"
      "THE ANGLE TO TIP IS BETWEEN  44 AND 110 DEGREES WITH RESPECT TO THE -X
       AXIS"
      "THE OUT AND BACK SIGNALS ARE THE SAME")
         :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THE OUT OR BACK SIGNALS DO NOT CURVE TOWARDS -X"
      "THE ANGLE TO TIP IS OUTSIDE 44 TO 110 DEGREES WITH RESPECT TO THE -X
       AXIS"
      "THE OUT AND BACK SIGNALS ARE DIFFERENT")
         :then 'od-analysis
         :then-english-template
      "THE EVENT IS DUE TO AN OUTER DIAMETER WALL LOSS OF ## %"))

; NON-NOSP
; PREDICATE THAT RETURNS T IF EVENT I IS NOT A NOSP
(defun non-nosp (structure i)
  (if (not (nth i (get-value structure nosp?)))
      T
      nil))

; CURVE-TO--X?
; PREDICATE THAT RETURNS T IF EITHER OUT OR BACK CURVES TO THE -X AXIS
; ACTUALLY THIS MEASURES WHETHER THE SIGNAL IS COUNTER-CLOCKWISE OR NOT
;  AND ONLY MEASURES CURVATURE TO -X IN THE SECOND QUADRANT
(defun curve-to--x? (structure i)
  (or (nth i (get-value structure curve-to--x-out?))
      (nth i (get-value structure curve-to--x-back?))))

; ANGLE-TO-TIP-OD?
; PREDICATE THAT RETURNS T IF THE ANGLE TO TIP IS IN BETWEEN 44 AND 110
(defun angle-to-tip-od? (structure i)
  (and (>= (setf angle (nth i (get-value structure tip-angle))) 44)
       (<= angle 110)))

; O=B?
; PREDICATE THAT RETURNS T IF THE NN OUT AND BACK SIGNALS ARE THE SAME
(defun o=b? (structure i)
  (equal (nth i (get-value structure nn-o=b?)) 'T))   ; COULD BE 'T OR 'F
```

```
; OD-ANALYSIS
; RETURNS THE WALL LOSS - A FUNCTION OF THE TIP ANGLE WRT -X AXIS
; WRITES THE WALL LOSS TO THE WALL LOSS SLOT IN THE TUBE STRUCTURE
; ONLY CALLED IF EVENT I PASSES THE RULE FOR OD WALL LOSS
(defun od-analysis (structure i)
  (let ((angle (nth i (get-value structure tip-angle)))
        (wall-loss-list (get-value structure wall-loss)))
    (cond ((and (>= angle 44) (<= angle 100))
           (setf wl (* 10 (round (/ (- 150 (* angle 1.25)) 10)))))
          ((and (> angle 100) (<= angle 108))
           (setf wl (* 10 (round (/ (- 212.5 (* angle 1.875)) 10)))))
          ((and (> angle 108) (<= angle 110))
           (setf wl 5)))
    (setf (nth i wall-loss-list) wl)
    (put-value structure wall-loss wall-loss-list)
    (string-append "OD - " (string-num wl) "%")))
```

```
; ID-ID
; NON-NOSP, CURVE-TO--X? AND O=B? ARE IN THE OD-ID.LSP FILE
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf id-id
      (make-rule
        :ifs '(non-nosp curve-to--x? angle-to-tip-id? o=b?)
        :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THE OUT OR BACK SIGNAL CURVES TOWARDS -X"
      "THE ANGLE TO TIP IS BETWEEN  15 AND 35 DEGREES WITH RESPECT TO THE -X
       AXIS"
      "THE OUT AND BACK SIGNALS ARE THE SAME")
        :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THE OUT OR BACK SIGNALS DO NOT CURVE TOWARDS -X"
      "THE ANGLE TO TIP IS OUTSIDE 15 TO 35 DEGREES WITH RESPECT TO THE -X
       AXIS"
      "THE OUT AND BACK SIGNALS ARE DIFFERENT")
        :then 'id-analysis
        :then-english-template
      "THE EVENT IS DUE TO AN INNER DIAMETER WALL LOSS OF ## %"))


; ANGLE-TO-TIP-ID?
; PREDICATE THAT RETURNS T IF THE TIP ANGLE IS BETWEEN 15 AND 35 DEGREES
;  WRT -X AXIS.
(defun angle-to-tip-id? (structure i)
  (and (>= (setf angle (nth i (get-value structure tip-angle))) 15)
       (<= angle 35)))

; ID-ANALYSIS
; RETURNS THE WALL LOSS - A FUNCTION OF THE TIP ANGLE WRT -X AXIS
; WRITES THE WALL LOSS TO THE WALL LOSS SLOT IN THE TUBE STRUCTURE
; ONLY CALLED IF EVENT I PASSES THE RULE FOR ID WALL LOSS
(defun id-analysis (structure i)
  (let ((angle (nth i (get-value structure tip-angle)))
        (wall-loss-list (get-value structure wall-loss)))
    (cond ((and (>= angle 15) (<= angle 18.125))
           (setf wl 10))
          ((and (> angle 18.125) (<= angle 35))
           (setf wl (* 25 (round (/ (- (* 4 angle) 60) 25))))))
    (setf (nth i wall-loss-list) wl)
    (put-value structure wall-loss wall-loss-list)
    (string-append "ID - " (string-num wl) "%")))
```

```
; HOLE-ID
; NON-NOSP, CURVE-TO--x? AND O=B? ARE IN OD-ID.LSP
(setf hole-id
      (make-rule
        :ifs '(non-nosp curve-to--x? angle-to-tip-hole? o=b?)
        :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THE OUT OR BACK SIGNAL CURVES TOWARDS -X"
      "THE ANGLE TO TIP IS BETWEEN  35 AND 44 DEGREES WITH RESPECT TO THE -X
       AXIS"
      "THE OUT AND BACK SIGNALS ARE THE SAME")
        :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THE OUT OR BACK SIGNALS DO NOT CURVE TOWARDS -X"
      "THE ANGLE TO TIP IS OUTSIDE 35 TO 44 DEGREES WITH RESPECT TO THE -X
       AXIS"
      "THE OUT AND BACK SIGNALS ARE DIFFERENT")
        :then 'hole-analysis
        :then-english-template
      "THE EVENT IS DUE TO A HOLE"))


; ANGLE-TO-TIP-HOLE?
; PREDICATE THAT RETURNS T IF THE ANGLE TO TIP IS IN BETWEEN 35 AND 44
;  DEGREES WRT -X AXIS
(defun angle-to-tip-hole? (structure i)
  (and (>= (setf angle (nth i (get-value structure tip-angle))) 35)
       (<= angle 44)))

; HOLE-ANALYSIS
; RETURNS THE WALL LOSS - A FUNCTION OF THE TIP ANGLE WRT -X AXIS
; WRITES THE WALL LOSS TO THE WALL LOSS SLOT IN THE TUBE STRUCTURE
; ONLY CALLED IF EVENT I PASSES THE RULE FOR HOLES
(defun hole-analysis (structure i)
  (let ((angle (nth i (get-value structure tip-angle)))
        (wall-loss-list (get-value structure wall-loss)))
    (setf (nth i wall-loss-list) 100)
    (put-value structure wall-loss wall-loss-list)
    "HOLE"))
```

```
; DENT-ID
; NON-NOSP AND O=B? ARE IN OD-ID.LSP
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf dent-id
       (make-rule
         :ifs '(non-nosp within-10? o=b? straight?)
         :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THE ANGLE TO TIP IS WITHIN + OR - 10 DEGREES OF THE X AXIS"
      "THE OUT AND BACK SIGNALS ARE THE SAME"
      "THE SIGNAL IS STRAIGHT")
         :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THE ANGLE TO TIP IS NOT WITHIN + OR - 10 DEGREES OF THE X AXIS"
      "THE OUT AND BACK SIGNALS ARE NOT THE SAME"
      "THE SIGNAL IS NOT STRAIGHT")
         :then 'dent-analysis
         :then-english-template
      "THE EVENT IS DUE TO A DENT"))

; WITHIN-10?
; PREDICATE THAT RETURNS T IF THE TIP ANGLE IS WITHIN 10 DEGREES OF THE X AXIS
(defun within-10? (structure i)
  (and (>= (setf angle                                    ; ANGLE IS MEASURED
                                                          ;  FROM -X AXIS
                 (nth i (get-value structure tip-angle))) 170)
       (<= angle 190)))

; STRAIGHT?
; PREDICATE THAT RETURNS T IF THE TRACE IS STRAIGHT
; "STRAIGHT" MEANS THAT THE AVERAGE OF THE BASIS ANGLES FOR THE OUT SIGNAL
;  IS WITHIN + OR - 2 DEGREES OF THE TIP ANGLE - O=B? TAKES CARE OF THE BACK
;  SIGNAL
; AVERAGE IS IN NOSP.LSP
; TRANS-TO--X IS IN NON-NOSP.LSP
; TO-DEGREES IS IN CALCPROP.LSP
(defun straight? (structure i)
  (and (>= (setf avg-angle-out
                 (to-degrees
                   (get-value structure angles)
                     (average
                       (trans-to--x
                          structure
                         (nth i (get-value structure nn-b-out))))))
           (- (setf angle (nth i (get-value structure tip-angle))) 2))
       (<= avg-angle-out (+ angle 2))))
```

```
; DENT-ANALYSIS
; ONLY CALLED IF EVENT I PASSES THE RULE FOR DENTS
(defun dent-analysis (structure i)
  "DENT")
```

```
; WHAL-ID
; THIS CONTAINS THE RULE FOR THE ID OF WHALIES
;  IT IS, HOWEVER INCOMPLETE BECAUSE THE SEPARATE X AND Y TRACES ARE NOT
;  EXAMINED FOR THE CHARACTERISTIC WIDTH OF PEAK THAT IS > 5 TIMES THE
;  NORMAL WIDTH OD SUPPORT PLATE TRACES
; NON-NOSP AND O=B? IS IN OD-ID.LSP AND STRAIGHT? IS IN DENT-ID.LSP
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf whal-id
       (make-rule
         :ifs '(non-nosp half-len-to-tip? straight? o=b? in-quad-2?)
         :ifs-english-true '(
       "THE EVENT IS NOT A NOSP"
       "THE LENGTH TO TIP IS < OR = HALF THAT OF THE AVERAGE NOSP LENGTH TO
        TIP"
       "THE SIGNAL IS STRAIGHT"
       "THE OUT AND BACK SIGNALS ARE THE SAME"
       "THE SIGNAL IS IN THE SECOND QUADRANT")
          :ifs-english-false '(
       "THE EVENT IS A NOSP"
       "THE LENGTH TO TIP IS > HALF THAT OF THE AVERAGE NOSP LENGTH TO
        TIP"
       "THE SIGNAL IS NOT STRAIGHT"
       "THE OUT AND BACK SIGNALS ARE DIFFERENT"
       "THE SIGNAL IS NOT IN THE SECOND QUADRANT")
          :THEN 'WHALIE-ANALYSIS
          :then-english-template
       "THE EVENT IS DUE TO A WHALIE"))

; HALF-LEN-TO-TIP?
; PREDICATE THAT RETURNS T IF THE LENGTH TO TIP IS <= THE AVERAGE NOSP
;  LENGTH TO TIP
(defun half-len-to-tip? (structure i)
  (and (<= (nth i (get-value structure tip-length))
           (* .5 (get-value structure nosp-avg-tip-length)))))

; IN-QUAD-2?
; PREDICATE THAT RETURNS T IF THE SIGNAL IS IN THE SECOND QUADRANT
; THE ANGLE IS WITH RESPECT TO THE -X AXIS
(defun in-quad-2? (structure i)
  (and (<= (setf angle (nth i (get-value structure tip-angle)))
           90)
       (>= angle 0)))

; WHALIE-ANALYSIS
; ONLY CALLED IF EVENT I PASSES THE RULE FOR WHALIES
(defun whalie-analysis (structure i)
  "WHALIE")
```

```
; WLSPC-ID
; WALL LOSS AT SUPPORT PLATE CENTER
; PROSP-TEST IS IN NOSP-ID.LSP AND NON-NOSP IS IN OD-ID.LSP
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf wlspc-id
      (make-rule
        :ifs '(prosp-test non-nosp longer-len-to-tip?)
        :ifs-english-true '(
     "THE SIGNAL IS A PROSP"
     "THE SIGNAL IS NOT A NOSP"
     "THE TIP LENGTH IS MORE THAN 10% LONGER THAN THE AVERAGE NOSP TIP
      LENGTH")
        :ifs-english-false '(
     "THE SIGNAL IS NOT A PROSP"
     "THE SIGNAL IS A NOSP"
     "THE TIP LENGTH IS NOT MORE THAN 10% LONGER THAN THE AVERAGE NOSP TIP
      LENGTH")
        :then 'wall-loss-at-sp-center
        :then-english-template
     "THERE IS WALL LOSS (## %) AT THE SUPPORT PLATE CENTER"))

; LONGER-LEN-TO-TIP?
; PREDICATE THAT RETURNS T IF THE TIP LENGTH IS MORE THAN 10% LONGER THAN
;  THE AVERAGE NOSP TIP LENGTH
(defun longer-len-to-tip? (structure i)
  (and (>= (nth i (get-value structure tip-length))
           (* 1.1 (get-value structure nosp-avg-tip-length)))))

; WALL-LOSS-AT-SP-CENTER
; RETURNS THE WALL LOSS - A FUNCTION OF THE TIP ANGLE WRT -X AXIS
; WRITES THE WALL LOSS TO THE WALL LOSS SLOT IN THE TUBE STRUCTURE
; ONLY CALLED IF EVENT I PASSES THE RULE FOR WALL LOSS AT THE SUPPORT
;  PLATE CENTER
(defun wall-loss-at-sp-center (structure i)
  (let* ((x (nth i (get-value structure x)))
         (y (nth i (get-value structure y)))
         (av-tip**2 (expt (get-value structure nosp-avg-tip-length) 2.)))
    (catch 'up
      (dotimes (j (setf tip-ind (nth i (get-value structure tip-index))))
        (when (>= (+ (expt (nth j x) 2.) (expt (nth j y) 2.))
                  av-tip**2)
              (setf save-j j)
              (throw 'up nil)))
      (setf save-j tip-ind))
   (if (not (equal tip-ind save-j))
       (progn (setf angle (rad-to-deg
                           (- (atan (- (nth tip-ind y) (nth save-j y))
```

```
                                        (- (nth tip-ind x) (nth save-j x))))))
              (string-append "sp-"
                             (sp-analysis angle)))
       ""))))     ; NO WALL LOSS AT SP

; RAD-TO-DEG
; CONVERTS RADIANS TO DEGREES
(defun rad-to-deg (angle)
  (round (* angle (/ 180 pi))))

; SP-ANALYSIS
; SEE OD-ID, ID-ID and HOLE-ID
; RETURNS WALL LOSS ANALYSIS AS A FUNCTION OF ANGLE
; CALLED BY WALL-LOSS-AT-SP-CENTER
(defun sp-analysis (angle)
  (cond ((and (>= angle 15) (<= angle 18.125))
         (setf wl 10)
         (string-append "ID-" (string-num wl) "%"))
        ((and (> angle 18.125) (<= angle 35))
         (setf wl (* 25 (round (/ (- (* 4 angle) 60) 25))))
         (string-append "ID-" (string-num wl) "%"))
        ((and (> angle 35) (< angle 44))
         "HOLE")
        ((and (>= angle 44) (<= angle 100))
         (setf wl (* 10 (round (/ (- 150 (* angle 1.25)) 10))))
         (string-append "OD-" (string-num wl) "%"))
        ((and (> angle 100) (<= angle 108))
         (setf wl (* 10 (round (/ (- 212.5 (* angle 1.875)) 10))))
         (string-append "OD-" (string-num wl) "%"))
        ((and (> angle 108) (<= angle 110))
         (setf wl 5)
         (string-append "OD-" (string-num wl) "%"))))
```

```
; SCALL-ID
; IDENTIFIES SCALLOP BARS
; NON-NOSP AND O=B? ARE IN OD-ID; IN-QUAD-2? IS IN WHAL-ID
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf scall-id
       (make-rule
         :ifs '(non-nosp scall-size? scall-angle? curve-ccw? o=b? in-quad-2?)
         :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THE TIP LENGTH IS 15% to 25% THAT OF THE AVERAGE NOSP TIP LENGTH"
      "THE TIP ANGLE IS -20 to -30 DEGREES LESS THAN THAT OF THE AVERAGE
       NOSP TIP ANGLE"
      "THE SIGNAL CURVES COUNTER-CLOCKWISE"
      "THE OUT AND BACK SIGNALS ARE THE SAME"
      "THE INITIAL ANGLE IS IN THE SECOND QUADRANT")
         :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THE TIP LENGTH IS NOT 15% TO 25% OF THAT OF THE AVERAGE NOSP TIP LENGTH"
      "THE TIP ANGLE IS NOT -20 TO -30 DEGREES LESS THAN THAT OF THE AVERAGE
       NOSP TIP ANGLE"
      "THE SIGNAL DOES NOT CURVE COUNTER-CLOCKWISE"
      "THE OUT AND BACK SIGNALS ARE DIFFERENT"
      "THE INITIAL ANGLE IS NOT IN THE SECOND QUADRANT")
         :then 'scallop-bar
         :then-english-template
      "THE EVENT IS DUE TO A SCALLOP BAR"))


; SCALL-SIZE?
; PREDICATE THAT RETURNS T IF THE TIP LENGTH IS BETWEEN 15% AND 25% OF
;  THE AVERAGE NOSP TIP LENGTH
(defun scall-size? (structure i)
  (and (>= (setf length (nth i (get-value structure tip-length)))
           (* .15 (setf avg-length (get-value structure nosp-avg-tip-length))))
       (<= length (* .25 avg-length))))

; SCALL-ANGLE?
; PREDICATE THAT RETURNS T IF TIP ANGLE IS 20 TO 30 DEGREES LESS THAN THE
;  AVERAGE NOSP TIP ANGLE
(defun scall-angle? (structure i)
  (and (>= (setf angle (nth i (get-value structure tip-angle)))
           (- (setf avg-angle (get-value structure nosp-avg-tip-angle)) 30))
       (<= angle (- avg-angle 20))))

; CURVE-CCW?
; PREDICATE THAT RETURNS T IF TRACE CURVES CCW
; ACTUALLY CURVE-TO--X? MEASURES WHETHER THE SIGNAL IS COUNTER-CLOCKWISE OR NOT
```

```
;  AND ONLY MEASURES CURVATURE TO -X IN THE SECOND QUADRANT - SO CURVE-TO--x?
;  CAN BE USED HERE
(defun curve-ccw? (structure i)
  (curve-to--x? structure i))

; SCALLOP-BAR
; ONLY CALLED IF EVENT I PASSES THE RULE FOR SCALLOP BARS
(defun scallop-bar (structure i)
  "sc bar")
```

```
; SAT-ID
; IDENTIFIES SATURATED SIGNALS
; NON-NOSP IS IN OD-ID.LSP
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf sat-id
      (make-rule
        :ifs '(non-nosp too-many-edge-values?)
        :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THERE ARE MORE THAN 20 VALUES OF ONE OF THE FOLLOWING: MAXIMUM X,
       MINIMUM X, MAXIMUM Y OR MINIMUM Y")
        :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THERE ARE NOT MORE THAN 20 VALUES OF ONE OF THE FOLLOWING:
       MAXIMUM X, MINIMUM X, MAXIMUM Y OR MINIMUM Y")
        :then 'saturation-analysis
        :then-english-template
      "THE SIGNAL IS SATURATED"))

; TOO-MANY-EDGE-VALUES?
; PREDICATE THAT RETURNS T IF THERE ARE MORE THAN 20 VALUES OF THE MAX +
;  (- MAX 1) + (- max 2) IN THE X OR Y DIRECTIONS; OR IF THERE ARE MORE
;  THAN 20 VALUES OF THE MIN + (+ MIN 1) + (+ MIN 2) IN THE X OR Y DIRECTIONS.
; OUR-APPLY IS IN RULE.LSP
(defun too-many-edge-values? (structure i)
  (let* ((max-x (our-apply #'max (setf x (nth i (get-value structure x)))))
         (min-x (our-apply #'min x))
         (max-y (our-apply #'max (setf y (nth i (get-value structure y)))))
         (min-y (our-apply #'min y)))
    (or (< (length (remove (- max-x 2)
                           (remove (- max-x 1) (remove max-x x)))) (- (length x) 20))
        (< (length (remove (+ min-x 2)
                           (remove (+ min-x 1) (remove min-x x)))) (- (length x) 20))
        (< (length (remove (- max-y 2)
                           (remove (- max-y 1) (remove max-y y)))) (- (length y) 20))
        (< (length (remove (+ min-y 2)
                           (remove (+ min-y 1) (remove min-y y)))) (- (length y) 20)))
    ))

; SATURATION-ANALYSIS
; ONLY CALLED IF EVENT I PASSES THE RULE FOR SATURATION
(defun saturation-analysis (structure i)
  " SAT")
```

```
; HAFSP-ID
; IDENTIFICATION OF HALF A SUPPORT PLATE SIGNAL FOR THE OUT OR BACK SIGNAL
; NON-NOSP IS IN OD-ID.LSP
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf hafsp-id
      (make-rule
        :ifs '(non-nosp ia-q-or? s-shape-or? min-x-angle? min-x-length?)
        :ifs-english-true '(
      "THE EVENT IS NOT A NOSP"
      "THE OUT OR BACK SIGNAL INITIAL ANGLE IS IN THE 1ST OR 2ND QUADRANT"
      "THE OUT OR BACK SIGNAL HAS AN S-SHAPE"
      "THE SIGNAL TIP ANGLE (MEASURED FROM THE MINIMUM X POINT) IS WITHIN
       nosp-angle DEGREES OF THE NOSP AVERAGE TIP ANGLE"
      "THE SIGNAL TIP LENGTH (MEASURED FROM THE MINIMUM X POINT) IS WITHIN
       nosp-length (FRACTION) OF THE NOSP AVERAGE TIP LENGTH")
        :ifs-english-false '(
      "THE EVENT IS A NOSP"
      "THE OUT OR BACK SIGNAL INITIAL ANGLE IS NOT IN THE 1ST OR 2ND QUADRANT"
      "THE OUT OR BACK SIGNAL DOES NOT HAVE AN S-SHAPE"
      "THE SIGNAL TIP ANGLE (MEASURED FROM THE MINIMUM X POINT) IS NOT
       WITHIN nosp-angle DEGREES OF THE NOSP AVERAGE TIP ANGLE"
      "THE SIGNAL TIP LENGTH (MEASURED FROM THE MINIMUM X POINT) IS NOT
       WITHIN nosp-length (FRACTION) OF THE NOSP AVERAGE TIP LENGTH")
        :then 'half-support-plate-analysis
        :then-english-template
      "THE EVENT IS DUE TO A HALF SUPPORT SIGNAL"))

; IA-Q-OR?
; PREDICATE THAT RETURNS T IF THE INITIAL ANGLE OUT OR BACK IS BETWEEN 0 AND
;  180 DEGREES WRT -X AXIS
; BETWEEN IS IN PROSP-ID.LSP
(defun ia-q-or? (structure i)
  (cond ((or (between (nth i
                      (get-value structure initial-angle-out)) 0 180)
             (between (nth i
                      (get-value structure initial-angle-back)) 0 180))
 T)
        (T nil)))


; S-SHAPE-OR?
; PREDICATE THAT RETURNS T IF EITHER THE OUT OR BACK SIGNALS HAVE AN S SHAPE
(defun s-shape-or? (structure i)
  (cond ((or (equal (nth i (get-value structure s-shape-out?)) 't)
             (equal (nth i (get-value structure s-shape-back?)) 't))
         T)
        (T nil)))
```

```
; MIN-X-ANGLE?
; PREDICATE THAT RETURNS T IF THE TIP ANGLE (DEFINED BY THE MINIMUM X VALUE)
;  IS WITHIN NOSP-ANGLE DEGRESS OF THE AVERAGE NOSP TIP ANGLE
; FIND-ANGLE IS IN CALCPROP.LSP
(defun min-x-angle? (structure i)
  (let* ((x (nth i (get-value structure x)))
         (y (nth i (get-value structure y)))
         (min-x (our-apply #'min x)))
    (if (< min-x 0)
        (progn
          (let* ((index-min-x (- (length x) (length (member min-x x))))
                 (angle-min-x (find-angle index-min-x x y))
                 (avg-angle (get-value structure nosp-avg-tip-angle))
                 (tol (get-value structure nosp-angle)))
            (if (between angle-min-x (- avg-angle tol) (+ avg-angle tol))
                T
                nil)))
        nil)))


; MIN-X-LENGTH?
; PREDICATE THAT RETURNS T IF THE TIP LENGTH (DEFINED BY THE MINIMUM X VALUE)
;  IS WITHIN NOSP-LENGTH OF THE AVERAGE NOSP TIP LENGTH
; FIND-LENGTH IS IN CALCPROP.LSP
(defun min-x-length? (structure i)
  (let* ((x (nth i (get-value structure x)))
         (y (nth i (get-value structure y)))
         (min-x (our-apply #'min x))
         (index-min-x (- (length x) (length (member min-x x))))
         (length-min-x (find-length index-min-x x y))
         (avg-length (get-value structure nosp-avg-tip-length))
         (tol (* (get-value structure nosp-length) avg-length)))
   (if (between length-min-x (- avg-length tol) (+ avg-length tol))
       T
       nil)))

; HALF-SUPPORT-PLATE-ANALYSIS
; ONLY CALLED IF EVENT I PASSES THE RULE FOR HALF SUPPORT PLATES
(defun half-support-plate-analysis (structure i)
  "HALF SP")
```

```
; SFRET-ID
; IDENTIFIES A SYMMETRIC FRET AT A SUPPORT PLATE
; NON-NOSP IS IN OD-ID.LSP
; IA-Q, O=B and S-SHAPE ARE IN PROSP-ID
; SEE RULE.LSP FOR RULE STRUCTURE AND COMMENTS
; RULE.LSP MUST BE COMPILED BEFORE THIS FILE
(setf sfret-id
      (make-rule
        :ifs '(any-nosps? non-nosp ia-q o=b s-shape rotated-up? length>=?)
        :ifs-english-true '(
      "THIS TUBE HAS SOME NORMAL SUPPORT PLATE SIGNALS"
      "THE SIGNAL IS NOT A NORMAL SUPPORT PLATE SIGNAL"
      "THE INITIAL ANGLE (BOTH OUT AND BACK) IS IN THE 1ST OR 2ND QUADRANT"
      "THE OUT AND BACK SIGNALS ARE THE SAME"
      "THE S SHAPE IS PRESENT IN BOTH THE OUT AND BACK SIGNALS"
      "THE SIGNAL IS ROTATED CLOCKWISE FROM THE NOSP AVERAGE TIP ANGLE"
      "THE LENGTH IS NOT LESS THAN THE NOSP AVERAGE TIP LENGTH")
        :ifs-english-false '(
      "THIS TUBE HAS NO NORMAL SUPPORT PLATE SIGNALS"
      "THIS SIGNAL IS DUE TO A NORMAL SUPPORT PLATE"
      "THE INITIAL ANGLE (BOTH OUT AND BACK) IS NOT IN THE 1ST OR 2ND QUADRANT"
      "THE OUT AND BACK SIGNALS ARE NOT THE SAME"
      "THE S SHAPE IS NOT PRESENT IN BOTH THE OUT AND BACK SIGNALS"
      "THE SIGNAL IS NOT ROTATED FROM THE NOSP AVERAGE TIP ANGLE"
      "THE LENGTH IS LESS THAN THE NOSP AVERAGE TIP LENGTH")
        :then 'symmetric-fret-analysis
        :then-english-template
      "THE EVENT IS DUE TO A SYMMETRIC FRET AT A SUPPORT PLATE WITH
       WALL LOSS OF ## %"))

; ANY-NOSPS?
; PREDICATE THAT RETURNS T IF THE TUBE HAS ANY NOSPS
(defun any-nosps? (structure i)
  (> (length (get-value structure nosp-events)) 0))

; ROTATED-UP?
; PREDICATE THAT RETURNS T IF THE TIP ANGLE IS ROTATED CW FROM THE
;  AVERAGE NOSP TIP ANGLE BY AT LEAST NOSP-ANGLE
(defun rotated-up? (structure i)
  (if (> (nth i (get-value structure tip-angle))
         (+ (get-value structure nosp-angle)
            (get-value structure nosp-avg-tip-angle)))
      T
      nil))


; LENGTH>=?
; PREDICATE THAT RETURNS T IF THE TIP LENGTH IS > THE AVERAGE NOSP TIP ANGLE
```

```
;  MINUS NOSP LENGTH
(defun length>=? (structure i)
  (if (> (nth i (get-value structure tip-length))
         (* (- 1 (get-value structure nosp-length))
            (get-value structure nosp-avg-tip-length)))
      T
      nil))

; SYMMETRIC-FRET-ANALYSIS
; SP-ANALYSIS AND RAD-TO-DEG ARE IN WLSPC-ID.LSP
; ONLY CALLED IF EVENT I PASSES THE RULE FOR SYMMETRIC FRETS
(defun symmetric-fret-analysis (structure i)
  (let* ((angle-avg (* (get-value structure nosp-avg-tip-angle) (/ pi 180.))
         (x-tip-avg (- (* (setf length
                                (get-value structure nosp-avg-tip-length))
                          (cos angle-avg))))
         (y-tip-avg (* length (sin angle-avg)))
         (x-tip (nth (setf index (nth i (get-value structure tip-index)))
                     (nth i (get-value structure x))))
         (y-tip (nth index
                     (nth i (get-value structure y)))))
    (setf angle (rad-to-deg (-                    ; SO ANGLE IS WRT -X AXIS
                             (atan (- y-tip y-tip-avg)
                                   (- x-tip x-tip-avg)))))
    (string-append "SF-" (sp-analysis angle))))
```

APPENDIX H

ENGLISH TRANSLATION OF THE RULES

NOTES:

1. THE SYMBOLS IN LOWER CASE REFER TO CONSTANTS IN THE TUBE DATA STRUCTURE.

2. THE NAME IN SQUARE BRACKETS AFTER THE RULE NUMBER IS THE NAME OF THE LISP SOURCE FILE WHERE THE RULE IS DEFINED AND DOCUMENTED. THE PREDICATES (SEE 3.) WILL BE IN THIS FILE OR A REFERENCE WILL BE MADE TO THE FILE THAT CONTAINS THEM.

3. THE NAME IN SQUARE BRACKETS AFTER EACH LINE IN THE IF LIST IS THE CORRESPONDING PREDICATE.

RULE 1 [PROSP-ID.LSP]

IF
THE TIP ANGLE IS WITHIN + OR - 20 DEGREES OF THE -X AXIS
[T-A20]
AND THE INITIAL ANGLE (BOTH OUT AND BACK) IS IN THE 1ST OR 2ND QUADRANT
[IA-Q]
AND THE OUT AND BACK SIGNALS ARE THE SAME
[O=B]
AND THE S-SHAPE IS PRESENT IN BOTH THE OUT AND BACK SIGNALS
[S-SHAPE]

THEN
THE EVENT IS PROVISIONALLY DUE TO A NORMAL SUPPORT PLATE (PROSP)
[PROSP-ANALYSIS]

\- - + - -

RULE 2 [NOSP-ID.LSP]

IF
THE EVENT IS A PROSP
[PROSP-TEST]
AND THE TIP LENGTH IS WITHIN nosp-length (FRACTION) OF THE PROSP AVERAGE TIP LENGTH
[PROSP-AVG-LEN]
AND THE TIP ANGLE IS WITHIN nosp-angle DEGREES OF THE PROSP AVERAGE TIP ANGLE
[PROSP-AVG-ANGLE]

```
 THEN
      THE EVENT IS DUE TO A NORMAL SUPPORT PLATE (NOSP)
      [NOSP-ANALYSIS]

                              - - + - -


RULE 3      [OD-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE OUT OR BACK SIGNAL CURVES TOWARDS -X
      [CURVE-TO--X?]
  AND THE ANGLE TO TIP IS BETWEEN  44 AND 110 DEGREES WITH RESPECT TO THE -X
      AXIS
      [ANGLE-TO-TIP-OD?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]

 THEN
      THE EVENT IS DUE TO AN OUTER DIAMETER WALL LOSS OF ## %
      [OD-ANALYSIS]

                              - - + - -


RULE 4      [ID-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE OUT OR BACK SIGNAL CURVES TOWARDS -X
      [CURVE-TO--X?]
  AND THE ANGLE TO TIP IS BETWEEN  15 AND 35 DEGREES WITH RESPECT TO THE -X
      AXIS
      [ANGLE-TO-TIP-ID?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]

 THEN
      THE EVENT IS DUE TO AN INNER DIAMETER WALL LOSS OF ## %
      [ID-ANALYSIS]

                              - - + - -


RULE 5      [HOLE-ID.LSP]
```

```
IF
     THE EVENT IS NOT A NOSP
     [NON-NOSP]
 AND THE OUT OR BACK SIGNAL CURVES TOWARDS -X
     [CURVE-TO--X?]
 AND THE ANGLE TO TIP IS BETWEEN  35 AND 44 DEGREES WITH RESPECT TO THE -X
     AXIS
     [ANGLE-TO-TIP-HOLE?]
 AND THE OUT AND BACK SIGNALS ARE THE SAME
     [O=B?]

THEN
     THE EVENT IS DUE TO A HOLE
     [HOLE-ANALYSIS]
```

- - + - -

```
RULE 6     [DENT-ID.LSP]

 IF
     THE EVENT IS NOT A NOSP
     [NON-NOSP]
  AND THE ANGLE TO TIP IS WITHIN + OR - 10 DEGREES OF THE X AXIS
     [WITHIN-10?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
     [O=B?]
  AND THE SIGNAL IS STRAIGHT
     [STRAIGHT?]

 THEN
     THE EVENT IS DUE TO A DENT
     [DENT-ANALYSIS]
```

- - + - -

```
RULE 7     [WHAL-ID.LSP]

 IF
     THE EVENT IS NOT A NOSP
     [NON-NOSP]
  AND THE LENGTH TO TIP IS < OR = HALF THAT OF THE AVERAGE NOSP LENGTH TO
     TIP
     [HALF-LEN-TO-TIP?]
  AND THE SIGNAL IS STRAIGHT
     [STRAIGHT?]
```

```
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]
  AND THE SIGNAL IS IN THE SECOND QUADRANT
      [IN-QUAD-2?]

 THEN
      THE EVENT IS DUE TO A WHALIE
      [WHALIE-ANALYSIS]
```

- - + - -

```
RULE 8     [WLSPC-ID.LSP]

 IF
      THE SIGNAL IS A PROSP
      [PROSP-TEST]
  AND THE SIGNAL IS NOT A NOSP
      [NON-NOSP]
  AND THE TIP LENGTH IS MORE THAN 10% LONGER THAN THE AVERAGE NOSP TIP
      LENGTH
      [LONGER-LEN-TO-TIP?]

 THEN
      THERE IS WALL LOSS (## %) AT THE SUPPORT PLATE CENTER
      [WALL-LOSS-AT-SP-CENTER]
```

- - + - -

```
RULE 9     [SCALL-ID.LSP]

 IF
      THE EVENT IS NOT A NOSP
      [NON-NOSP]
  AND THE TIP LENGTH IS 15% to 25% THAT OF THE AVERAGE NOSP TIP LENGTH
      [SCALL-SIZE?]
  AND THE TIP ANGLE IS -20 to -30 DEGREES LESS THAN THAT OF THE AVERAGE
      NOSP TIP ANGLE
      [SCALL-ANGLE?]
  AND THE SIGNAL CURVES COUNTER-CLOCKWISE
      [CURVE-CCW?]
  AND THE OUT AND BACK SIGNALS ARE THE SAME
      [O=B?]
  AND THE INITIAL ANGLE IS IN THE SECOND QUADRANT
      [IN-QUAD-2?]

 THEN
```

THE EVENT IS DUE TO A SCALLOP BAR
[SCALLOP-BAR]

- - + - -

RULE 10 [SAT-ID.LSP]

IF
THE EVENT IS NOT A NOSP
[NON-NOSP]
AND THERE ARE MORE THAN 20 VALUES OF ONE OF THE FOLLOWING: MAXIMUM X, MINIMUM X, MAXIMUM Y OR MINIMUM Y
[TOO-MANY-EDGE-VALUES?]

THEN
THE SIGNAL IS SATURATED
[SATURATION-ANALYSIS]

- - + - -

RULE 11 [HAFSP-ID.LSP]

IF
THE EVENT IS NOT A NOSP
[NON-NOSP]
AND THE OUT OR BACK SIGNAL INITIAL ANGLE IS IN THE 1ST OR 2ND QUADRANT
[IA-Q-OR?]
AND THE OUT OR BACK SIGNAL HAS AN S-SHAPE
[S-SHAPE-OR?]
AND THE SIGNAL TIP ANGLE (MEASURED FROM THE MINIMUM X POINT) IS WITHIN nosp-angle DEGREES OF THE NOSP AVERAGE TIP ANGLE
[MIN-X-ANGLE?]
AND THE SIGNAL TIP LENGTH (MEASURED FROM THE MINIMUM X POINT) IS WITHIN nosp-length (FRACTION) OF THE NOSP AVERAGE TIP LENGTH
[MIN-X-LENGTH?]

THEN
THE EVENT IS DUE TO A HALF SUPPORT SIGNAL
[HALF-SUPPORT-PLATE-ANALYSIS]

- - + - -

RULE 12 [SFRET-ID.LSP]

IF

```
     THIS TUBE HAS SOME NORMAL SUPPORT PLATE SIGNALS
     [ANY-NOSPS?]
 AND THE SIGNAL IS NOT A NORMAL SUPPORT PLATE SIGNAL
     [NON-NOSP]
 AND THE INITIAL ANGLE (BOTH OUT AND BACK) IS IN THE 1ST OR 2ND QUADRANT
     [IA-Q]
 AND THE OUT AND BACK SIGNALS ARE THE SAME
     [O=B]
 AND THE S SHAPE IS PRESENT IN BOTH THE OUT AND BACK SIGNALS
     [S-SHAPE]
 AND THE SIGNAL IS ROTATED CLOCKWISE FROM THE NOSP AVERAGE TIP ANGLE
     [ROTATED-UP?]
 AND THE LENGTH IS NOT LESS THAN THE NOSP AVERAGE TIP LENGTH
     [LENGTH>=?]

THEN
     THE EVENT IS DUE TO A SYMMETRIC FRET AT A SUPPORT PLATE WITH
     WALL LOSS OF ## %
     [SYMMETRIC-FRET-ANALYSIS]
```

- - + - -

APPENDIX I

INFERENCE ENGINE FILES

```
; THIS CODE IS (MOSTLY) COPIED FROM: LISP, 2ND EDITION BY WINSTON AND HORN
;  (ADDISON-WESLEY 1984), CHAPTERS 17 AND 18.

; FORWARD-CHAIN
; STEPS THROUGH THE RULE LIST; EACH TIME A NEW ASSERTION IS FOUND,
;  IT STARTS AT THE BEGINNING OF THE LIST
; THE CODE IN THE BOOK DIDN'T WORK PROPERLY
(defun forward-chain ()
  (do ((rules-to-try  rules)                    ; NOTE DIFFERENCE FROM BOOK
       (progress-made nil))
      ((null rules-to-try) progress-made)
     (cond ((use-rule (car rules-to-try))
            (setq rules-to-try rules)
            (setq progress-made t))
           (t (setq rules-to-try
                    (cdr rules-to-try))))))  ; NOTE DIFFERENCE FROM BOOK

; USE-RULE
; TAKES A RULE AS ARGUMENT
; GIVES THE IF PART TO CASCADE-THROUGH-PATTERNS (IN REVERSE ORDER -
;  SEE CAS...) AND GIVES THE RESULTING A-LIST-STREAM AND THE THE PART TO
;  FEED-TO-ACTIONS
; IF THERE ARE NO WAYS TO MATCH THE ANTECEDENTS TO THE DATA OR IF THERE ARE
;  NO NEW WAYS THEN USE-RULE RETURNS NIL; OTHERWISE T
(defun use-rule (rule)
  (let* ((rule-name (cadr rule))
         (ifs (reverse (cdr (caddr rule))))
         (thens (cdr (car (cdddr rule))))           ; NOTE: BOOK HAS CADDDR
         (a-list-stream (cascade-through-patterns
                         ifs
                         (add-to-stream nil         ; THIS PRODUCES '(NIL)
                                                    ;  NIL WOULD MEAN THAT
                                                    ;  THE PROGRAM WOULD
                                                    ;  NEVER PASS LINE 2 OF
                                                    ;  FILTER-A-LIST-STREAM
                                        (make-empty-stream))))
         (action-stream (feed-to-actions rule-name thens a-list-stream)))
         (not (empty-stream-p action-stream))))

; FEED-TO-ACTIONS
; SPREAD THROUGH ACTIONS ACCEPTS ONLY A-LISTS; THIS PROCEDURE
;  ACCEPTS AN A-LIST-STREAM AND FEEDS IT TO SPREAD-THROUGH-ACTIONS ONE
;  A-LIST AT A TIME, COMBINING THE RESULTING ACTION STREAMS INTO ONE STREAM
(defun feed-to-actions (rule-name actions a-list-stream)
  (cond ((empty-stream-p a-list-stream) (make-empty-stream))
        (t (combine-streams
            (spread-through-actions rule-name
                                    actions
```

```
                                    (first-of-stream a-list-stream))
            (feed-to-actions rule-name
                             actions
                             (rest-of-stream a-list-stream))))))

; REPLACE-VARIABLES
; REPLACE THE < VARIABLES WITH ITEMS FROM A-LIST
(defun replace-variables (s a-list)
  (cond ((atom s) s)                                    ; ATOM -> ATOM
        ((equal (car s) 'call) s)                       ; DEMON -> (CALL ..)
        ((equal (car s) '<)                             ; (< VAR)=>REPLACE VAR
         (cadr (assoc (pattern-variable s) a-list)))
        (t (cons (replace-variables (car s) a-list)
                 (replace-variables (cdr s) a-list)))))

; SPREAD-THROUGH-ACTIONS
; THIS PROCEDURE TAKES THE LIST OF ACTIONS IN ACTIONS, REPLACES
;  THE < VARIABLES WITH VALUES FROM A-LIST, TRIES TO ADD THE RESULTING
;  ASSERTIONS TO THE DATA AND RETURNS A LIST OF THE NEW ASSERTIONS
(defun spread-through-actions (rule-name actions a-list)
  (do ((actions actions (cdr actions))                  ; VARIABLE 1
       (action-stream (make-empty-stream)))             ; VARIABLE 2
      ((null actions) action-stream)                    ; END TEST
    (let ((action (replace-variables (car actions)      ; NEW ASSERTION
                                     a-list)))
      (cond ((remember action)                          ; TRY TO SAVE
             (demon action)                             ; IF ACTION IS IN THE
                                                        ;  DEMON FORMAT CALL A
                                                        ;  PROCEDURE
;           (print '(rule ,rule-name says ,@action))
            (setq action-stream (add-to-stream action   ; SAVE FOR RETURN
                                               action-stream)))))))

; DEMON
; NOT IN BOOK
; FOR ACTIONS WITH THE FORMAT:
;    ((CALL FUNCTION) WITH ARGUEMENTS: ARGUMENT(S))
; CALLS THE FUNCTION FUNCTION WITH ARGUMENTS ARGUMENTS
(defun demon (action)
  (when (and (listp (car action))
             (equal (caar action) 'call))
        (apply (cadar action) (nthcdr 3 action)))
  action)

; CASCADE-THROUGH-PATTERNS
; FILTER-A-LIST-STREAM ACCEPTS ONE ANTECEDENT WHEREAS WE NEED TO PROCESS
;  A LIST OF ANTECEDENTS
; CASCADE THROUGH PATTERNS DOES THIS PASSING THE OUTPUT OF ONE TO THE INPUT
```

```
;  OF ANOTHER
; NOTE THAT IT DOES THE LAST ON FIRST SO THAT THE IF PARTS OF RULES SHOULD
;  BE IN REVERSE ORDER
; FOR FIRST CALL A-LIST STREAM SHOULD BE '(NIL)
; PATTERNS CONTAINS A LIST OF ANTECEDENTS AND A-LIST-STREAM CONTAINS A LIST
;  OF A-LISTS OF V-V PAIRS
; RETURNS A STREAM OF A-LISTS
(defun cascade-through-patterns (patterns a-list-stream)
  (cond ((null patterns) a-list-stream)
        (t (filter-a-list-stream (car patterns)
                                 (cascade-through-patterns (cdr patterns)
                                                           a-list-stream)))))

; FILTER-A-LIST-STREAM
; FILTER-ASSERTIONS ACCEPTS AN A-LIST WHERAS IN GENERAL WE WISH TO FEED
;  A STREAM OF A-LISTS TO FILTER ASSERTIONS
; FILTER-A-LIST-STREAM DOES THIS AND RETURNS A STREAM OF A-LISTS
; EXAMPLE: (FILTER-A-LIST-STREAM '((<ANIMAL) IS A PARENT OF (> CHILD)
;                                '(((ANIMAL SWEEKUMS) (TYPE PENGUIN))
;                                  ((ANIMAL BOZO) (TYPE CHEETAH))))
;          -> (((ANIMAL BOZO) (TYPE CHEETAH) (CHILD BILLY))
;              ((ANIMAL BOZO) (TYPE CHEETAH) (CHILD SUGAR)))
(defun filter-a-list-stream (pattern a-list-stream)
  (cond ((empty-stream-p a-list-stream) (make-empty-stream))
        (t (combine-streams
            (filter-assertions pattern (first-of-stream a-list-stream))
            (filter-a-list-stream pattern (rest-of-stream a-list-stream))))))

; FILTER-ASSERTIONS
; PATTERN IS AN ANTECEDENT TAKEN FROM SOME RULE
; INITIAL-A-LIST IS AN A-LIST OF VARIABLE-VALUE PAIRS
; ASSERTIONS IS THE GLOBAL VARIABLE CARRYING THE INITIAL AND DERIVED FACTS
; FOR EACH ITEM IN ASSERTIONS, FILTER-ASSERTIONS MATCHES TO PATTERN
;  (USING THE VARIABLE-VALUE PAIRS IN INITIAL-A-LIST FOR THE (< VARIABLE)
;  MATCHES) RETURNING AN A-LIST OF V-V PAIRS.
; THE STREAM OF NEW A-LISTS (OF V-V PAIRS) (IE A LIST OF LISTS OF V-V PAIRS)
;  IS RETURNED
; EXAMPLE: (FILTER-ASSERTIONS '((> ANIMAL) IS A (> TYPE)) NIL)
;          -> (((ANIMAL SWEEKUMS) (TYPE PENGUIN))
;              ((ANIMAL BOZO) (TYPE CHEETAH)))
; EXAMPLE: (FILTER-ASSERTIONS '((< ANIMAL) IS A PARENT OF (> CHILD))
;                             '((ANIMAL BOZO) (TYPE CHEETAH)))
;          -> (((ANIMAL BOZO) (TYPE CHEETAH) (CHILD BILLY))
;              ((ANIMAL BOZO) (TYPE CHEETAH) (CHILD SUGAR)))
(defun filter-assertions (pattern initial-a-list)
  (do ((assertions assertions (cdr assertions))          ; FIRST VAR BINDING
       (a-list-stream (make-empty-stream)))              ; SECOND VAR BINDING
      ((null assertions) a-list-stream)                  ; END TEST
```

```
   (let ((new-a-list (match pattern (car assertions)    ; MATCH RETURNS NEW
                                    initial-a-list)))   ;  V-V PAIRS
     (cond (new-a-list (setq a-list-stream              ; ADD TO A-LIST-STREAM
                             (add-to-stream new-a-list a-list-stream))))))

; EMPTY-STREAM-P
(defun empty-stream-p (s) (null s))

; MAKE-EMPTY-STREAM
(defun make-empty-stream () nil)

; REST-OF-STREAM
(defun rest-of-stream (s) (cdr s))

; FIRST-OF-STREAM
(defun first-of-stream (s) (car s))

; ADD-TO-STREAM
(defun add-to-stream (e s) (cons e s))

; COMBINE-STREAMS
(defun combine-streams (s1 s2) (append s1 s2))

; REMEMBER
; ADD A NEW ASSERTION TO ASSERTIONS
(defun remember (new)
  (cond ((member new assertions :test 'equal)   ; IF PRESENT DO NOTHING
         nil)
        (t (setq assertions                     ; ADD NEW TO ASSERTIONS
                 (cons new assertions))
           new)))                               ; RETURN NEW ASSERTION

(setq assertions
      '((bozo is a cheetah)
        (bozo is a parent of sugar)
        (bozo is a parent of billy)
        (sweekums is a penguin)
        (king is a parent of rex)
        (sweekums is a parent of rover)
        (rover is a parent of spot)))

(setq rules '((rule identify16
               (if ((> animal) is a (> type))
                   ((< animal) is a parent of (> child)))
               (then ((< child) is a (< type))
                     ((call animal) with arguements: (< type) (< child) 3)))))

(defun animal (type child number)
```

```
  (print '(type ,type child ,child number ,number))
  (setf aaa (* number 3)))


(defun fc ()
  (forward-chain))

;-------------------------------------------------------------------------------
;                              PATTERN MATCHING PROGRAMS                        |
;-------------------------------------------------------------------------------

; MATCHER
; GETS RID OF THE NIL ARGUMENT OF MATCH
(defun matcher (p d)
   (match p d nil))


; MATCH
; MATCH MATCHES THE PATTERN IN P TO THE DATUM IN D.
; ? IN P MATCHES ANY ATOM IN D AND + MATCHES ANY NUMBER OF ATOMS.
; (> VARIABLE1)  IN P (LIKE ?) CAN MATCH A SINGLE ATOM IN D AND
;   (+ VARIABLE2) IN P (LIKE +) CAN MATCH ANY NUMBER OF ATOMS IN D.
;   THE DIFFERENCE IS THAT VARIABLE1 TAKES THE VALUE OF THE SINGLE ATOM
;   AND VARIABLE2 BECOMES A LIST OF THE ATOMS THAT IT MATCHED.
; ASSIGNMENTS CONTAINS AN ASSOCIATION LIST OF VARIABLE-VALUE
;   PAIRS CONTAINING MATCHES OF THE (> VAR) AND (+ VAR) VARIABLES.
; (< VARIABLE) CAUSES A SEARCH OF ASSIGNMENTS FOR VARIABLE.  IF IT IS
;   FOUND THEN THE VALUE IS PULLED OUT FOR THE MATCH.
; (RESTRICT ? PREDICATE-1 .. PREDICATE-N) CAUSES THE EQUIVALENT ATOM
;   IN D TO BE TESTED WITH PREDICATE PREDICATE-1 .. PREDICATE-N.
; IF THERE ARE NO PATTERN VARIABLES THEN T IS RETURNED.  IF THERE
;   IS NO MATCH THEN NIL IS RETURNED.
; THE INITIAL VALUE OF ASSIGNMENTS IS NIL. - SEE MATCHER, ABOVE
; EXAMPLE: (MATCH '((> X) QUICK (+ Y)) '(THE QUICK BROWN FOX) NIL)
;          -> ((X THE) (Y (BROWN FOX)))
;
; OPTIONS IN P SUMMARY:
;
;  ATOM              MATCH ATOM
;  ?                 MATCH ANY ATOM
;  +                 MATCH ANY NUMBER OF ATOMS
;  (> VARIABLE)      MATCH ANY ATOM AND ASSOCIATE IT WITH VARIABLE
;  (+ VARIABLE)      MATCH ANY NUMBER OF ATOMS AMD ASSOCIATE IT WITH VARIABLE
;  (< VARIABLE)      PULL VARIABLE FROM A-LIST AND MATCH IT TO ATOM IN D
;  (RESTRICT ?       RESTRICT A MATCH TO AN ATOM IN D TO THOSE FOR WHICH ALL
;     PREDICATE-1      THE INDICATED PREDICATES ARE TRUE
;     .. PREDICATE-N)
(defun match (p d assignments)
```

```
(cond ((and (null p) (null d))                    ; BOTH P AND D ARE EMPTY
       (cond ((null assignments) t)               ; RETURN T IF NO ASSIGNMENTS
             (t assignments)))                    ; RETURN ASSIGNMENTS
      ((or (null p) (null d)) nil)                ; ONE SHORTER -> NIL
      ((or (equal (car p) '?)                     ; ? IN P MATCHES D
           (equal (car p) (car d)))               ; 1ST ELEMENTS EQUAL
       (match (cdr p) (cdr d) assignments))       ; MATCH REST OF LISTS
      ((equal (car p) '+)                         ; MATCH + PATTERN
       (or (match (cdr p) (cdr d) assignments);   SEE IF REST MATCHES
           (match p (cdr d) assignments)))        ; KEEP +
      ((atom (car p)) nil)                        ; ALREADY TESTED TO SEE IF
                                                  ; CARs MATCH.  NOW CHECK FOR
                                                  ; (> ..) (+ ..) OR (< ..)
      ((equal (pattern-indicator (car p)) '>) ; LOOK FOR (> ..) IN (CAR P)
       (match (cdr p) (cdr d)                     ; IF REST OF P & D MATCH
              (shove-gr                           ; THEN GIVE THE PATTERN
                 (pattern-variable (car p))       ; VARIABLE THE VALUE OF
                 (car d)                          ; (CAR D)
                 assignments)))
      ((equal (pattern-indicator (car p)) '<) ; LOOK FOR (< ..) IN (CAR P)
       (match (cons (pull-value                   ; PULL MATCHING VALUE FROM
                      (pattern-variable           ; ASSIGNMENTS, ATTACH TO FRONT
                        (car p)) assignments)     ; OF (CDR P), THEN MATCH THE
                    (cdr p))                      ; RESULT TO D
              d
              assignments))
      ((equal (pattern-indicator (car p)) '+) ; LOOK FOR (+ ..) IN (CAR P)
       (let ((new-assignments
              (shove-pl                           ; PUT VARIABLE ON ASSIGNMENTS
                        (pattern-variable (car p))
                        (car d)
                        assignments)))
            (or (match (cdr p)                    ; MATCH (CDR P) WITH (CDR D)
                       (cdr d) new-assignments)
                (match p                          ; MORE TO (+ ..) MATCH
                       (cdr d) new-assignments))))
      ((and (equal (pattern-indicator (car p))
                   'restrict)                     ; CHECK FOR RESTRICT
            (equal (restriction-indicator         ; COULD PUT OTHER TYPES IN
                       (car p)) '?)
            (test (restriction-predicates         ; SEE IF PREDICATES ARE TRUE
                       (car p)) (car d)))
       (match (cdr p) (cdr d) assignments))))

; PATTERN-INDICATOR
; CALLED BY MATCH
; 1 HAS THE FORM (> VAR) (< VAR) OR (+ VAR)
; ->  >, < OR +
```

```
(defun pattern-indicator (l)
  (car l))


; PATTERN-VARIABLE
; CALLED BY MATCH
; l HAS THE FORM (> VAR) (< VAR) OR (+ VAR)
; -> VAR
(defun pattern-variable (l)
  (cadr l))


; SHOVE-GR
; PUT A NEW VARIABLE-ITEM PAIR ON THE ASSIGNMENTS A-LIST - MATCHING (> ..)
(defun shove-gr (variable item a-list)
  (append a-list (list (list variable item))))

; SHOVE-PL
; PUT A NEW VARIABLE-LIST PAIR ON THE ASSIGNMENTS A-LIST - MATCHING (+ ..)
; MORE COMPLICATED THAN SHOVE-GR BECAUSE IT MUST CHECK TO SEE IF A LIST
; HAS ALREADY BEEN STARTED
(defun shove-pl (variable item a-list)
  (cond ((null a-list)                          ; A-LIST EMPTY?
         (list (list variable (list item))))    ; YES => START A-LIST
        ((equal variable (caar a-list))         ; VAR 1ST OF A-LIST?
         (cons (list variable                   ; YES => ADD ITEM TO
                     (append (cadar a-list)     ;  VARIABLE'S LIST
                             (list item)))
               (cdr a-list)))
        (t (cons (car a-list)                   ; CHECK REST OF A-LIST FOR VAR
                 (shove-pl variable item (cdr a-list))))))

; PULL-VALUE
; CALLED BY MATCH
; EXTRACTS VALUE OF VARIABLE FROM A-LIST
(defun pull-value (variable a-list)
  (cadr (assoc variable a-list)))

; RESTRICTION-INDICATOR
; PATTERN-ITEM HAS FORM (RESTRICT ? PREDICATE-1 ..)
; -> ?
(defun restriction-indicator (pattern-item) (cadr pattern-item))

; RESTRICTION-PREDICATES
; PATTERN-ITEM HAS FORM (RESTRICT ? PREDICATE-1 ..)
; -> (PREDICATE-1 PREDICATE-2 ..)
(defun restriction-predicates (pattern-item) (cddr pattern-item))
```

```
; TEST
; PREDICATES HAS FORM (PREDICATE-1 PREDICATE-2 ..)
(defun test (predicates argument)
  (cond ((null predicates) t)
        ((funcall (car predicates) argument)
         (test (cdr predicates) argument))     ; CONTINUE IF 1ST ONE TRUE
        (t nil)))
```

APPENDIX J

EXPECT VARIABLES FOR TUBES T-CAL-0, T-1-22R, AND T-2-6

```
FILE NAME  T-CAL-0
TUBE NUMBER  calibration
STATION
UNIT  6
HEAT EXCHANGER TYPE  Moderator Hx
HEAT EXCHANGER NUMBER  2
FREQUENCY  220k
DATE
TIME
SUPERVISOR  eg
COMMENT
AVERAGE TIP LENGTH  0
AVERAGE TIP ANGLE  0
NORMAL SUPPORT PLATES  NIL

FILE NAME  T-CAL-0
EVENT  1
TIP-INDEX  56
TIP-ANGLE  116
TIP-LENGTH  48.7545
VECTOR-ENDS-OUT  (0 4 13 21 27 38 49)
VECTOR-ENDS-BACK  (206 139 128 113 103 95 89 81 73)
BASIS-OUT    (6 14 16 13 14 18 11)
             (- - - - - - -)
BASIS-BACK   (0 2 12 8 16 17 7 22 14)
             (- - - - - - - - -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  34
INITIAL-ANGLE-BACK 0
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?  NIL
NN-VE-OUT  (0 4 13 21 27 38 49)
NN-VE-BACK  (206 139 128 113 103 95 89 81 73)
NN-B-OUT  (6 14 16 13 14 18 11)
NN-B-BACK  (0 2 12 8 16 17 7 22 14)
NN-BL-OUT  7
NN-BL-BACK  9
NN-IA-OUT  34
NN-IA-BACK  0
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
```

```
ANALYSIS   ( SAT)



FILE NAME  T-CAL-0
EVENT  2
TIP-INDEX  63
TIP-ANGLE  107
TIP-LENGTH  47.1275
VECTOR-ENDS-OUT  (0 10 15 25 31 39 54)
VECTOR-ENDS-BACK  (171 152 143 125 116 110 99 92 76)
BASIS-OUT    (7 16 16 14 20 12 16)
             (- - - - - - -)
BASIS-BACK   (0 13 8 10 24 10 16 16 16)
             (- - - - - - - - -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  39
INITIAL-ANGLE-BACK 0
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?  NIL
NN-VE-OUT  (0 10 15 25 31 39 54)
NN-VE-BACK  (171 152 143 125 116 110 99 92 76)
NN-B-OUT  (7 16 16 14 20 12 16)
NN-B-BACK  (0 13 8 10 24 10 16 16 16)
NN-BL-OUT  7
NN-BL-BACK  9
NN-IA-OUT  39
NN-IA-BACK  0
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-0
EVENT  3
TIP-INDEX  54
TIP-ANGLE  21
TIP-LENGTH  44.0114
VECTOR-ENDS-OUT  (0 7 16 26 38 54)
```

```
VECTOR-ENDS-BACK  (235 185 154 131 124 109 98 90 80 73 54)
BASIS-OUT     (23 27 36 34 32 34)
              (- - + + + +)
BASIS-BACK    (4 20 20 30 30 30 28 34 32 34 30)
              (- - - - - - - - + + + -)
BASIS-LENGTH-OUT  6
BASIS-LENGTH-BACK 11
INITIAL-ANGLE-OUT  129
INITIAL-ANGLE-BACK 22
S-SHAPE-OUT?  (2 2)
S-SHAPE-BACK? (7 7)
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 7 16 26 38 54)
NN-VE-BACK  (235 185 154 131 124 109 98 90 80 73 54)
NN-B-OUT  (23 27 36 34 32 34)
NN-B-BACK  (4 20 20 30 30 30 28 34 32 34 30)
NN-BL-OUT  6
NN-BL-BACK  11
NN-IA-OUT  129
NN-IA-BACK  22
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-0
EVENT  4
TIP-INDEX  34
TIP-ANGLE  37
TIP-LENGTH  33.6006
VECTOR-ENDS-OUT  (0 7 16)
VECTOR-ENDS-BACK  (109 90 79 69 60 47)
BASIS-OUT     (24 28 32)
              (- - +)
BASIS-BACK    (30 18 24 24 29 28)
              (- - - - - - -)
BASIS-LENGTH-OUT  3
BASIS-LENGTH-BACK 6
INITIAL-ANGLE-OUT  135
INITIAL-ANGLE-BACK 169
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
```

```
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 7 16)
NN-VE-BACK  (109 90 79 69 60 47)
NN-B-OUT  (24 28 32)
NN-B-BACK  (30 18 24 24 29 28)
NN-BL-OUT  3
NN-BL-BACK  6
NN-IA-OUT  135
NN-IA-BACK  169
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-0
EVENT  5
TIP-INDEX  55
TIP-ANGLE  108
TIP-LENGTH  45.2217
VECTOR-ENDS-OUT  (0 9 15 22 31 41)
VECTOR-ENDS-BACK  (163 139 125 113 107 100 95 85 77 67)
BASIS-OUT    (8 16 14 11 18 14)
             (- - - - - -)
BASIS-BACK   (56 8 12 8 12 14 18 16 14 18)
             (* - - - - - - - - - -)
BASIS-LENGTH-OUT  6
BASIS-LENGTH-BACK 10
INITIAL-ANGLE-OUT  45
INITIAL-ANGLE-BACK 315
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 9 15 22 31 41)
NN-VE-BACK  (163 139 125 113 107 100 95 85 77 67)
NN-B-OUT  (8 16 14 11 18 14)
NN-B-BACK  (56 8 12 8 12 14 18 16 14 18)
NN-BL-OUT  6
NN-BL-BACK  10
NN-IA-OUT  45
NN-IA-BACK  315
```

```
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-O
EVENT  6
TIP-INDEX  19
TIP-ANGLE  -6
TIP-LENGTH  27.1662
VECTOR-ENDS-OUT  (0 10)
VECTOR-ENDS-BACK  (67 61 53 49 42 36 25)
BASIS-OUT    (31 40)
             (- +)
BASIS-BACK   (57 28 28 32 30 30 36)
             (* - - + - - +)
BASIS-LENGTH-OUT  2
BASIS-LENGTH-BACK 7
INITIAL-ANGLE-OUT  174
INITIAL-ANGLE-BACK 321
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 10)
NN-VE-BACK  (67 61 53 49 42 36 25)
NN-B-OUT  (31 40)
NN-B-BACK  (57 28 28 32 30 30 36)
NN-BL-OUT  2
NN-BL-BACK  7
NN-IA-OUT  174
NN-IA-BACK  321
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL



FILE NAME  T-CAL-O
EVENT  7
```

```
TIP-INDEX  19
TIP-ANGLE  -6
TIP-LENGTH  27.1662
VECTOR-ENDS-OUT  (0 10)
VECTOR-ENDS-BACK  (67 61 53 49 42 36 25)
BASIS-OUT    (31 40)
             (- +)
BASIS-BACK   (57 28 28 32 30 30 36)
             (* - - + - - +)
BASIS-LENGTH-OUT  2
BASIS-LENGTH-BACK 7
INITIAL-ANGLE-OUT  174
INITIAL-ANGLE-BACK 321
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 10)
NN-VE-BACK  (67 61 53 49 42 36 25)
NN-B-OUT  (31 40)
NN-B-BACK  (57 28 28 32 30 30 36)
NN-BL-OUT  2
NN-BL-BACK  7
NN-IA-OUT  174
NN-IA-BACK  321
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL



FILE NAME  T-CAL-0
EVENT  8
TIP-INDEX  52
TIP-ANGLE  107
TIP-LENGTH  38.6005
VECTOR-ENDS-OUT  (0 5 11 23 33)
VECTOR-ENDS-BACK  (187 172 166 151 134 116 105 92 82 64)
BASIS-OUT    (9 14 18 12 19)
             (- - - - -)
BASIS-BACK   (55 18 62 20 12 14 18 14 10 19)
             (* - * - - - - - - -)
BASIS-LENGTH-OUT  5
BASIS-LENGTH-BACK 10
```

```
INITIAL-ANGLE-OUT  51
INITIAL-ANGLE-BACK 309
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 5 11 23 33)
NN-VE-BACK  (187 172 166 151 134 116 105 92 82 64)
NN-B-OUT  (9 14 18 12 19)
NN-B-BACK  (55 18 62 20 12 14 18 14 10 19)
NN-BL-OUT  5
NN-BL-BACK  10
NN-IA-OUT  51
NN-IA-BACK  309
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-0
EVENT  9
TIP-INDEX  74
TIP-ANGLE  104
TIP-LENGTH  49.4773
VECTOR-ENDS-OUT  (0 3 19 26 36 45 57)
VECTOR-ENDS-BACK  (184 160 148 139 130 122 114 103 92)
BASIS-OUT    (11 9 13 20 14 18 15)
             (- - - - - - - -)
BASIS-BACK   (16 2 16 14 8 18 16 16 16)
             (- - - - - - - - - -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  62
INITIAL-ANGLE-BACK 90
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 3 19 26 36 45 57)
NN-VE-BACK  (184 160 148 139 130 122 114 103 92)
NN-B-OUT  (11 9 13 20 14 18 15)
NN-B-BACK  (16 2 16 14 8 18 16 16 16)
```

```
NN-BL-OUT  7
NN-BL-BACK  9
NN-IA-OUT  62
NN-IA-BACK  90
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-O
EVENT  10
TIP-INDEX  35
TIP-ANGLE  50
TIP-LENGTH  34.0588
VECTOR-ENDS-OUT  (0 9 17)
VECTOR-ENDS-BACK  (145 111 91 81 72 64 54 43)
BASIS-OUT    (19 30 30)
             (- - -)
BASIS-BACK   (5 19 18 24 28 28 29 30)
             (- - - - - - - - -)
BASIS-LENGTH-OUT  3
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  107
INITIAL-ANGLE-BACK 28
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 9 17)
NN-VE-BACK  (145 111 91 81 72 64 54 43)
NN-B-OUT  (19 30 30)
NN-B-BACK  (5 19 18 24 28 28 29 30)
NN-BL-OUT  3
NN-BL-BACK  8
NN-IA-OUT  107
NN-IA-BACK  28
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)
```

```
FILE NAME  T-CAL-O
EVENT  11
TIP-INDEX  40
TIP-ANGLE  24
TIP-LENGTH  43.8634
VECTOR-ENDS-OUT  (0 5 12 18 21 28 36)
VECTOR-ENDS-BACK  (134 101 85 78 69 64 59 51 42)
BASIS-OUT    (21 26 28 32 36 29 36)
             (- - - + + - +)
BASIS-BACK   (16 20 26 25 32 30 33 30 34)
             (- - - - + - + - +)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  118
INITIAL-ANGLE-BACK 90
S-SHAPE-OUT?  (3 4)
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 5 12 18 21 28 36)
NN-VE-BACK  (134 101 85 78 69 64 59 51 42)
NN-B-OUT  (21 26 28 32 36 29 36)
NN-B-BACK  (16 20 26 25 32 30 33 30 34)
NN-BL-OUT  7
NN-BL-BACK  9
NN-IA-OUT  118
NN-IA-BACK  90
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-O
EVENT  12
TIP-INDEX  13
TIP-ANGLE  25
TIP-LENGTH  39.8121
VECTOR-ENDS-OUT  (0)
VECTOR-ENDS-BACK  (129 83 76 59 48 35)
BASIS-OUT    (28)
             (-)
```

```
BASIS-BACK   (13 19 28 34 30 34)
             (- - - + - +)
BASIS-LENGTH-OUT  1
BASIS-LENGTH-BACK 6
INITIAL-ANGLE-OUT  158
INITIAL-ANGLE-BACK 73
S-SHAPE-OUT?  F
S-SHAPE-BACK? (3 3)
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0)
NN-VE-BACK  (129 83 76 59 48 35)
NN-B-OUT  (28)
NN-B-BACK  (13 19 28 34 30 34)
NN-BL-OUT  1
NN-BL-BACK  6
NN-IA-OUT  158
NN-IA-BACK  73
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL



FILE NAME  T-CAL-0
EVENT  13
TIP-INDEX  68
TIP-ANGLE  108
TIP-LENGTH  47.4342
VECTOR-ENDS-OUT  (0 11 16 24 30 41 58)
VECTOR-ENDS-BACK  (162 145 135 124 118 106 97 87)
BASIS-OUT    (8 14 18 8 22 12 19)
             (- - - - - - -)
BASIS-BACK   (5 10 14 14 14 16 16 12)
             (- - - - - - - -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  45
INITIAL-ANGLE-BACK 28
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  T
PROSP?  NIL
NOSP?   NIL
```

```
NN-VE-OUT  (0 11 16 24 30 41 58)
NN-VE-BACK  (162 145 135 124 118 106 97 87)
NN-B-OUT  (8 14 18 8 22 12 19)
NN-B-BACK  (5 10 14 14 14 16 16 12)
NN-BL-OUT  7
NN-BL-BACK  8
NN-IA-OUT  45
NN-IA-BACK  28
NN-O=B?  T
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  10
ANALYSIS   (OD - 10%  SAT)



FILE NAME  T-CAL-0
EVENT  14
TIP-INDEX  33
TIP-ANGLE  46
TIP-LENGTH  30.4138
VECTOR-ENDS-OUT  (0 6 19 24 32)
VECTOR-ENDS-BACK  (139 110 102 91 78 65 55 47)
BASIS-OUT    (21 28 25 46 18)
             (- - - + -)
BASIS-BACK   (6 13 28 18 30 25 36 22)
             (- - - - - - + -)
BASIS-LENGTH-OUT  5
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  118
INITIAL-ANGLE-BACK 34
S-SHAPE-OUT?  (3 3)
S-SHAPE-BACK? (6 6)
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 6 19 24 32)
NN-VE-BACK  (139 110 102 91 78 65 55 47)
NN-B-OUT  (21 28 25 46 18)
NN-B-BACK  (6 13 28 18 30 25 36 22)
NN-BL-OUT  5
NN-BL-BACK  8
NN-IA-OUT  118
NN-IA-BACK  34
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
```

```
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-CAL-0
EVENT  15
TIP-INDEX  68
TIP-ANGLE  17
TIP-LENGTH  48.0833
VECTOR-ENDS-OUT  (0 11 20 29 38 47)
VECTOR-ENDS-BACK  (221 186 152 139 125 114 106 92 79)
BASIS-OUT    (24 32 30 30 28 39)
             (- + - - - +)
BASIS-BACK   (18 24 26 30 26 34 32 32 36)
             (- - - - - + + + +)
BASIS-LENGTH-OUT  6
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  135
INITIAL-ANGLE-BACK 101
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 11 20 29 38 47)
NN-VE-BACK  (221 186 152 139 125 114 106 92 79)
NN-B-OUT  (24 32 30 30 28 39)
NN-B-BACK  (18 24 26 30 26 34 32 32 36)
NN-BL-OUT  6
NN-BL-BACK  9
NN-IA-OUT  135
NN-IA-BACK  101
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL
```

```
FILE NAME  T-1-22R
TUBE NUMBER  1 22r
STATION
UNIT  6
HEAT EXCHANGER TYPE  Moderator Hx
HEAT EXCHANGER NUMBER  2
FREQUENCY  220k
DATE
TIME
SUPERVISOR  eg
COMMENT
AVERAGE TIP LENGTH  181.963
AVERAGE TIP ANGLE  -2.75
NORMAL SUPPORT PLATES  (1 2 3 4)

FILE NAME  T-1-22R
EVENT  1
TIP-INDEX  67
TIP-ANGLE  -3
TIP-LENGTH  185.219
VECTOR-ENDS-OUT  (16 21 25 29 33 37 49)
VECTOR-ENDS-BACK  (116 111 107 104 100 96 87)
BASIS-OUT    (23 32 35 36 35 33 29)
             (- + + + + + -)
BASIS-BACK   (24 33 34 35 36 34 31)
             (- + + + + + -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 7
INITIAL-ANGLE-OUT  129
INITIAL-ANGLE-BACK 135
S-SHAPE-OUT?  (1 3)
S-SHAPE-BACK? (1 4)
OUT=BACK?  T
PROSP?  T
NOSP?  T
ANALYSIS  sp



FILE NAME  T-1-22R
EVENT  2
TIP-INDEX  76
TIP-ANGLE  -2
TIP-LENGTH  177.138
VECTOR-ENDS-OUT  (15 21 25 29 33 39 66)
VECTOR-ENDS-BACK  (133 128 124 120 116 111)
```

```
BASIS-OUT     (20 34 35 37 35 33 28)
              (- + + + + + -)
BASIS-BACK    (25 32 34 35 36 35)
              (- + + + + +)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 6
INITIAL-ANGLE-OUT  112
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (1 3)
S-SHAPE-BACK? (1 4)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-22R
EVENT  3
TIP-INDEX  68
TIP-ANGLE  -3
TIP-LENGTH  182.222
VECTOR-ENDS-OUT  (17 22 26 29 33 37 51)
VECTOR-ENDS-BACK  (114 109 105 102 98 94 85)
BASIS-OUT     (22 34 35 36 35 33 28)
              (- + + + + + -)
BASIS-BACK    (25 32 34 36 36 35 30)
              (- + + + + + -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 7
INITIAL-ANGLE-OUT  124
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (1 3)
S-SHAPE-BACK? (1 3)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-22R
EVENT  4
TIP-INDEX  68
TIP-ANGLE  -3
TIP-LENGTH  183.273
```

```
VECTOR-ENDS-OUT  (18 22 26 29 33 38 51)
VECTOR-ENDS-BACK  (116 111 107 104 100 95 83)
BASIS-OUT     (24 33 36 36 35 32 30)
              (- + + + + + -)
BASIS-BACK    (25 33 34 36 36 35 29)
              (- + + + + + -)
BASIS-LENGTH-OUT  7
BASIS-LENGTH-BACK 7
INITIAL-ANGLE-OUT  135
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (1 2)
S-SHAPE-BACK? (1 3)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-1-22R
EVENT  5
TIP-INDEX  91
TIP-ANGLE  45
TIP-LENGTH  360.624
VECTOR-ENDS-OUT  (0 19 20 21 23 27 29 42 43 44)
VECTOR-ENDS-BACK  (184 183 182)
BASIS-OUT     (24 16 16 16 16 16 16 16 16 16)
              (- - - - - - - - - -)
BASIS-BACK    (32 22 16)
              (+ - -)
BASIS-LENGTH-OUT  10
BASIS-LENGTH-BACK 3
INITIAL-ANGLE-OUT  135
INITIAL-ANGLE-BACK 180
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0 19 20 21 22 24 26 27 28 29 30 42 43 44)
NN-VE-BACK  (184 183 182)
NN-B-OUT  (24 16 16 16 16 16 16 16 16 16 16 16 16 16)
NN-B-BACK  (32 22 16)
NN-BL-OUT  14
NN-BL-BACK  3
NN-IA-OUT  135
NN-IA-BACK  180
```

```
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  NIL
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-1-22R
EVENT  6
TIP-INDEX  4
TIP-ANGLE  -180
TIP-LENGTH  255.0
VECTOR-ENDS-OUT  (0)
VECTOR-ENDS-BACK  (23 20 18 16 14 12 10 8)
BASIS-OUT    (0)
             (-)
BASIS-BACK   (0 0 0 0 0 0 0 0)
             (- - - - - - - -)
BASIS-LENGTH-OUT  1
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  0
INITIAL-ANGLE-BACK 0
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0)
NN-VE-BACK  (24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7)
NN-B-OUT  (0)
NN-B-BACK  (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
NN-BL-OUT  1
NN-BL-BACK  18
NN-IA-OUT  0
NN-IA-BACK  0
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  NIL
WALL-LOSS  NIL
ANALYSIS   ( SAT)
```

```
FILE NAME  T-2-6
TUBE NUMBER  2 6
STATION
UNIT  6
HEAT EXCHANGER TYPE  Moderator Hx
HEAT EXCHANGER NUMBER  2
FREQUENCY  220k
DATE
TIME
SUPERVISOR  eg
COMMENT
AVERAGE TIP LENGTH  187.975
AVERAGE TIP ANGLE  -2.0
NORMAL SUPPORT PLATES  (2 3 4 5 7 8)

FILE NAME  T-2-6
EVENT  1
TIP-INDEX  24
TIP-ANGLE  59
TIP-LENGTH  63.2851
VECTOR-ENDS-OUT  (19 20 21)
VECTOR-ENDS-BACK  (33 30)
BASIS-OUT    (17 20 32)
             (- - +)
BASIS-BACK   (26 19)
             (- -)
BASIS-LENGTH-OUT  3
BASIS-LENGTH-BACK 2
INITIAL-ANGLE-OUT  96
INITIAL-ANGLE-BACK 146
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?  NIL
NN-VE-OUT  (5 16 18 19 20 21)
NN-VE-BACK  (55 40 39 38 34 33 32 31 30 28 27)
NN-B-OUT  (63 17 19 20 20 32)
NN-B-BACK  (53 23 29 14 40 25 19 21 17 23 18)
NN-BL-OUT  6
NN-BL-BACK  11
NN-IA-OUT  354
NN-IA-BACK  298
NN-O=B?  F
CURVE-TO--X-OUT?  T
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
```

```
ANALYSIS   NIL



FILE NAME  T-2-6
EVENT  2
TIP-INDEX  68
TIP-ANGLE  -2
TIP-LENGTH  199.123
VECTOR-ENDS-OUT  (12 21 25 28 31 34 37 41 55)
VECTOR-ENDS-BACK  (124 112 108 105 102 99 96 93 88)
BASIS-OUT    (24 27 34 35 36 35 35 32 27)
             (- - + + + + + + -)
BASIS-BACK   (28 26 31 33 34 35 37 36 32)
             (- - - + + + + + +)
BASIS-LENGTH-OUT  9
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  135
INITIAL-ANGLE-BACK 158
S-SHAPE-OUT?  (2 4)
S-SHAPE-BACK? (3 6)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-2-6
EVENT  3
TIP-INDEX  66
TIP-ANGLE  0
TIP-LENGTH  195.003
VECTOR-ENDS-OUT  (12 21 25 28 31 34 37 41 47)
VECTOR-ENDS-BACK  (117 109 105 102 99 96 93 89)
BASIS-OUT    (23 29 32 36 35 36 35 33 29)
             (- - + + + + + + -)
BASIS-BACK   (24 30 30 33 35 35 35 34)
             (- - - + + + + +)
BASIS-LENGTH-OUT  9
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  129
INITIAL-ANGLE-BACK 135
S-SHAPE-OUT?  (2 3)
S-SHAPE-BACK? (3 4)
OUT=BACK?  T
```

```
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-2-6
EVENT  4
TIP-INDEX  70
TIP-ANGLE  -1
TIP-LENGTH  173.012
VECTOR-ENDS-OUT  (14 19 22 25 28 31 34 39)
VECTOR-ENDS-BACK  (124 120 116 113 110 107 103 97)
BASIS-OUT    (21 30 34 36 35 36 34 31)
             (- - + + + + + -)
BASIS-BACK   (23 33 32 34 35 36 34 33)
             (- + + + + + + +)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  118
INITIAL-ANGLE-BACK 129
S-SHAPE-OUT?  (2 3)
S-SHAPE-BACK? (1 5)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-2-6
EVENT  5
TIP-INDEX  70
TIP-ANGLE  -3
TIP-LENGTH  177.181
VECTOR-ENDS-OUT  (16 20 23 26 29 32 36 49)
VECTOR-ENDS-BACK  (125 120 117 114 111 108 104 98)
BASIS-OUT    (23 33 34 36 36 34 32 28)
             (- + + + + + + -)
BASIS-BACK   (26 30 31 35 36 35 36 32)
             (- - - + + + + +)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 8
INITIAL-ANGLE-OUT  129
INITIAL-ANGLE-BACK 146
S-SHAPE-OUT?  (1 3)
```

```
S-SHAPE-BACK? (3 4)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-2-6
EVENT  6
TIP-INDEX  0
TIP-ANGLE  -4
TIP-LENGTH  153.395
VECTOR-ENDS-OUT  (0)
VECTOR-ENDS-BACK  (18 14 10 7 4 1)
BASIS-OUT    (33)
             (+)
BASIS-BACK   (25 30 32 34 35 36)
             (- - + + + +)
BASIS-LENGTH-OUT  1
BASIS-LENGTH-BACK 6
INITIAL-ANGLE-OUT  186
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0)
NN-VE-BACK  (28 22 18 16 14 13 11 9 8 7 6 5 4 3 2 1 0)
NN-B-OUT  (33)
NN-B-BACK  (20 27 28 27 31 31 33 33 33 36 34 36 36 37 34 36 34)
NN-BL-OUT  1
NN-BL-BACK  17
NN-IA-OUT  186
NN-IA-BACK  112
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL



FILE NAME  T-2-6
EVENT  7
```

```
TIP-INDEX  63
TIP-ANGLE  -4
TIP-LENGTH  193.437
VECTOR-ENDS-OUT  (14 20 23 26 29 31 34 37 42)
VECTOR-ENDS-BACK  (110 105 101 98 96 94 91 88 78)
BASIS-OUT    (21 31 34 37 35 36 36 32 32)
             (- - + + + + + + +)
BASIS-BACK   (25 30 33 34 37 36 37 35 28)
             (- - + + + + + + -)
BASIS-LENGTH-OUT  9
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  118
INITIAL-ANGLE-BACK 141
S-SHAPE-OUT?  (2 3)
S-SHAPE-BACK? (2 4)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp



FILE NAME  T-2-6
EVENT  8
TIP-INDEX  71
TIP-ANGLE  -2
TIP-LENGTH  190.095
VECTOR-ENDS-OUT  (12 19 23 26 29 32 35 39)
VECTOR-ENDS-BACK  (125 121 118 115 113 110 107 103 88)
BASIS-OUT    (20 30 35 35 37 35 35 33)
             (- - + + + + + +)
BASIS-BACK   (24 31 32 35 35 36 35 34 27)
             (- - + + + + + + -)
BASIS-LENGTH-OUT  8
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  112
INITIAL-ANGLE-BACK 135
S-SHAPE-OUT?  (2 4)
S-SHAPE-BACK? (2 5)
OUT=BACK?  T
PROSP?  T
NOSP?   T
ANALYSIS   sp
```

```
FILE NAME  T-2-6
EVENT  9
TIP-INDEX  18
TIP-ANGLE  -135
TIP-LENGTH  59.397
VECTOR-ENDS-OUT  (0)
VECTOR-ENDS-BACK  (56 46 45 43 20)
BASIS-OUT    (56)
             (*)
BASIS-BACK   (21 0 50 52 55)
             (- - * * *)
BASIS-LENGTH-OUT  1
BASIS-LENGTH-BACK 5
INITIAL-ANGLE-OUT  315
INITIAL-ANGLE-BACK 118
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0)
NN-VE-BACK  (94 93 88 79 75 73 71 56 50 47 46 45 44 43 42 39 20)
NN-B-OUT  (56)
NN-B-BACK  (2 16 26 16 27 14 47 22 20 1 60 50 55 16 51 57 58)
NN-BL-OUT  1
NN-BL-BACK  17
NN-IA-OUT  315
NN-IA-BACK  11
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  NIL
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-2-6
EVENT  10
TIP-INDEX  30
TIP-ANGLE  36
TIP-LENGTH  138.809
VECTOR-ENDS-OUT  (19 20 21 25 28 29)
VECTOR-ENDS-BACK  (43 41 40 35 34 33 32)
BASIS-OUT    (20 22 32 57 21 32)
             (- - + * - +)
BASIS-BACK   (29 12 49 33 25 23 26)
             (- - * + - - -)
```

```
BASIS-LENGTH-OUT  6
BASIS-LENGTH-BACK 7
INITIAL-ANGLE-OUT  112
INITIAL-ANGLE-BACK 163
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  T
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (6 11 15 19 20 21 24 25 27 28 29)
NN-VE-BACK  (55 51 48 46 45 44 43 42 41 40 39 36 35 34 33 32 30)
NN-B-OUT  (19 32 29 17 22 32 16 63 15 32 32)
NN-B-BACK  (16 17 29 9 55 37 22 25 6 49 46 28 28 25 23 26 0)
NN-BL-OUT  11
NN-BL-BACK  17
NN-IA-OUT  107
NN-IA-BACK  90
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   NIL



FILE NAME  T-2-6
EVENT  11
TIP-INDEX  16
TIP-ANGLE  -155
TIP-LENGTH  281.824
VECTOR-ENDS-OUT  (0)
VECTOR-ENDS-BACK  (206 197 196 195 194 190 189 35 34 33 32)
BASIS-OUT    (60)
             (*)
BASIS-BACK   (16 16 16 16 16 16 16 16 16 0 0)
             (- - - - - - - - - - -)
BASIS-LENGTH-OUT  1
BASIS-LENGTH-BACK 11
INITIAL-ANGLE-OUT  338
INITIAL-ANGLE-BACK 90
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (0)
NN-VE-BACK  (206 197 196 195 194 193 190 189 188 35 34 33 32)
```

```
NN-B-OUT  (60)
NN-B-BACK  (16 16 16 16 16 16 16 16 16 16 16 0 0)
NN-BL-OUT  1
NN-BL-BACK  13
NN-IA-OUT  338
NN-IA-BACK  90
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  NIL
WALL-LOSS  NIL
ANALYSIS   ( SAT)



FILE NAME  T-2-6
EVENT  12
TIP-INDEX  120
TIP-ANGLE  -135
TIP-LENGTH  360.624
VECTOR-ENDS-OUT  (19 21)
VECTOR-ENDS-BACK  (220 219 217 216)
BASIS-OUT    (16 0)
             (- -)
BASIS-BACK   (57 16 32 0)
             (* - + -)
BASIS-LENGTH-OUT  2
BASIS-LENGTH-BACK 4
INITIAL-ANGLE-OUT  90
INITIAL-ANGLE-BACK 321
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (19 21)
NN-VE-BACK  (220 219 217 216)
NN-B-OUT  (16 0)
NN-B-BACK  (57 16 32 0)
NN-BL-OUT  2
NN-BL-BACK  4
NN-IA-OUT  90
NN-IA-BACK  321
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  T
WALL-LOSS  NIL
ANALYSIS   ( SAT)
```

```
FILE NAME  T-2-6
EVENT  13
TIP-INDEX  50
TIP-ANGLE  -180
TIP-LENGTH  255.0
VECTOR-ENDS-OUT  (19 20 22 23)
VECTOR-ENDS-BACK  (94 92 90 88 86 84 82 80 78)
BASIS-OUT    (0 0 0 0)
             (- - - -)
BASIS-BACK   (0 0 0 0 0 0 0 0 0)
             (- - - - - - - - -)
BASIS-LENGTH-OUT  4
BASIS-LENGTH-BACK 9
INITIAL-ANGLE-OUT  0
INITIAL-ANGLE-BACK 0
S-SHAPE-OUT?  F
S-SHAPE-BACK? F
OUT=BACK?  F
PROSP?  NIL
NOSP?   NIL
NN-VE-OUT  (19 20 21 22 23)
NN-VE-BACK  (96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77)
NN-B-OUT  (0 0 32 0 0)
NN-B-BACK  (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
NN-BL-OUT  5
NN-BL-BACK  20
NN-IA-OUT  0
NN-IA-BACK  0
NN-O=B?  F
CURVE-TO--X-OUT?  NIL
CURVE-TO--X-BACK?  NIL
WALL-LOSS  NIL
ANALYSIS   ( SAT)
```

DISTRIBUTION[1]

Dr. S.B. Nickerson (3)

Public Reference Centre
Editorial Section (2)
Office Copies (2)
Extra Copies (3)

---

1. For Part II of Research Report No 87-173